

NAR-7090 Series Communication Appliance

User's Manual

Revision: 1.4

CE

This certificate of conformity of NAR-7090 series with actual required safety standards in accordance with 89/366 ECC-EMC Directive and LVD 73/23 ECC

UL

This product meets all safety requirements per UL60950 standard.



Portwell Inc.

4F, No.186, Jian-Yi RD., Chung-Ho City, 235 Taipei, Taiwan.
Headquarter: +886-2-7731-8888 FAX: +886-2-82271109
<http://www.portwell.com.tw>
Email: info@mail.portwell.com.tw



Table of Contents

Chapter 1	Introduction	3
1.1	About This Manual	3
1.2	Manual Organization	3
1.3	Technical Support Information	3
1.4	Board Layout.....	4
1.5	System Block Diagram.....	5
1.6	Product Specifications	7
1.7	LED Signaling Standard	8
Chapter 2	Getting Started.....	10
2.1	Included Hardware	10
2.2	Before You Begin	10
2.3	Hardware Configuration Setting	11
2.4	The Chassis	13
2.5	Open the Chassis.....	13
2.6	Install a Different Processor	14
2.7	Remove and Install DIMM.....	16
2.8	Remove and Install Compact Flash Card.....	18
2.9	Remove and Install Battery	18
2.10	Install HDD	19
2.11	Install Riser Card.....	19
2.12	Ear Mount Kit Installation	20
2.13	Remove EZIO / LCD	20
2.14	Remove Power Supply.....	22
2.15	Remove main board.....	24
2.16	Use a Client Computer.....	26
Chapter 3	BIOS Setting.....	29
Chapter 4	Programming Guide	38
4.1	Reset to Default Information	38
4.2	Bypass WDT Programming Guide	43
4.2.1	Bypass State Diagram	44
4.2.2	System flow description:	45
4.3	About EZIO2	59
4.4	GPIO Sample Code	70
4.5	WDT Sample Code	75
Chapter 5	Appendixes	103

5.1	<i>NAR-7090 Ethernet modules configuration.....</i>	103
-----	---	-----

Chapter 1 Introduction

1.1 About This Manual

This manual contains all required information for setting up and using the NAR- 7090 series.

NAR-7090 provides the essential platform for delivering optimal performance and functionality in the value communications appliance market segment. This manual should familiarize you with NAR-7090 operations and functions. NAR-7090 series provide up to eight on-board Ethernet ports to serve communication applications like Firewall, requiring ten Ethernet ports to connect external network (internet), demilitarized zone and internal network.

NAR-7090 series overview:

- ◆ Supports LG771 Intel 51XX series CPU
- ◆ Up to 32GB ECC/Register FB DIMM 533/667MHz
- ◆ Two USB ports and one RJ45 port on COM1.
- ◆ Four SATA connectors for SATA Hard disk
- ◆ User-friendly LCD control panel
- ◆ PCI-E architecture with totally three PCI-Ex8 interfaces.
- ◆ Provides absolute high flexibility of customized I/O configuration for front accessible PCI-E modules
--Maximum twelve PCI-E x4 GbE ports

1.2 Manual Organization

This manual describes how to configure your NAR-7090 system to meet various operating requirements. It is divided into three chapters, with each chapter addressing the basic concept and operation of this system.

- Chapter 1: Introduction. This section describes how this document is organized. It includes brief guidelines and overview to help find necessary information.
- Chapter 2: Hardware Configuration Setting and Installation. This chapter demonstrated the hardware assembly procedure, including detailed information. It shows the definitions and locations of Jumpers and Connectors that can be used to configure the system.
- Chapter 3: Operation Information. This section provides illustrations and information on the system architecture and how to optimize its performance.
- Chapter 4: This section describes how to programming software. It includes By-pass; GPIO; EZIO; Factory Default function.

Any updates to this manual, would be posted on the web site: <http://isc.portwell.com.tw>

1.3 Technical Support Information

Users may find helpful tips or related information on Portwell's web site: <http://www.portwell.com.tw>
A direct contact to Portwell's technical person is also available. For further support, users may also contact Portwell's headquarter in Taipei or local distributors.

1.4 Board Layout



Figure 1-1 Board Layout of NAR-7090 M/B

1.5 System Block Diagram

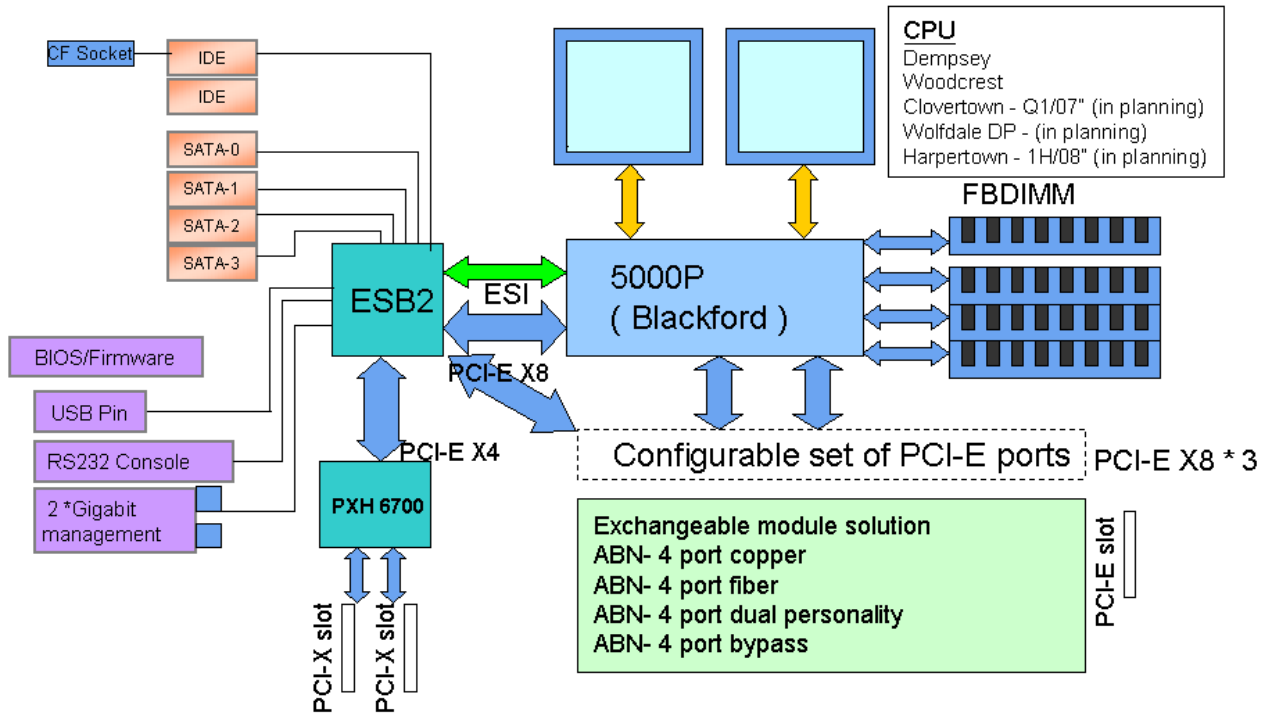


Figure 1-2 NAR-7090 Basic Block Diagram

NAR-7090 Series (Standard model)			
Sub-Model	NAR-7090-1013	NAR-7090-1012	NAR-7090-1017
Barebone #			
ATO #			
Slot-A1+B1	8 * Fiber	8* Copper with segment bypass	4* Fiber + 4* Copper
Slot-C1	ATO Option	ATO Option	ATO Option
Bypass Segment	N/A	4	N/A
Low profile PCI-X	1	1	1
PCI-E X4 slot or 64bit PCI-X Slot	Optional		
HDD	Standard 1* 3.5" HDD tray		
LCM	EZIO-3		
USB	2		
Console	RJ45 on COM1		
ATO-Options	CPU, FBDIMM HDD (SATA)		

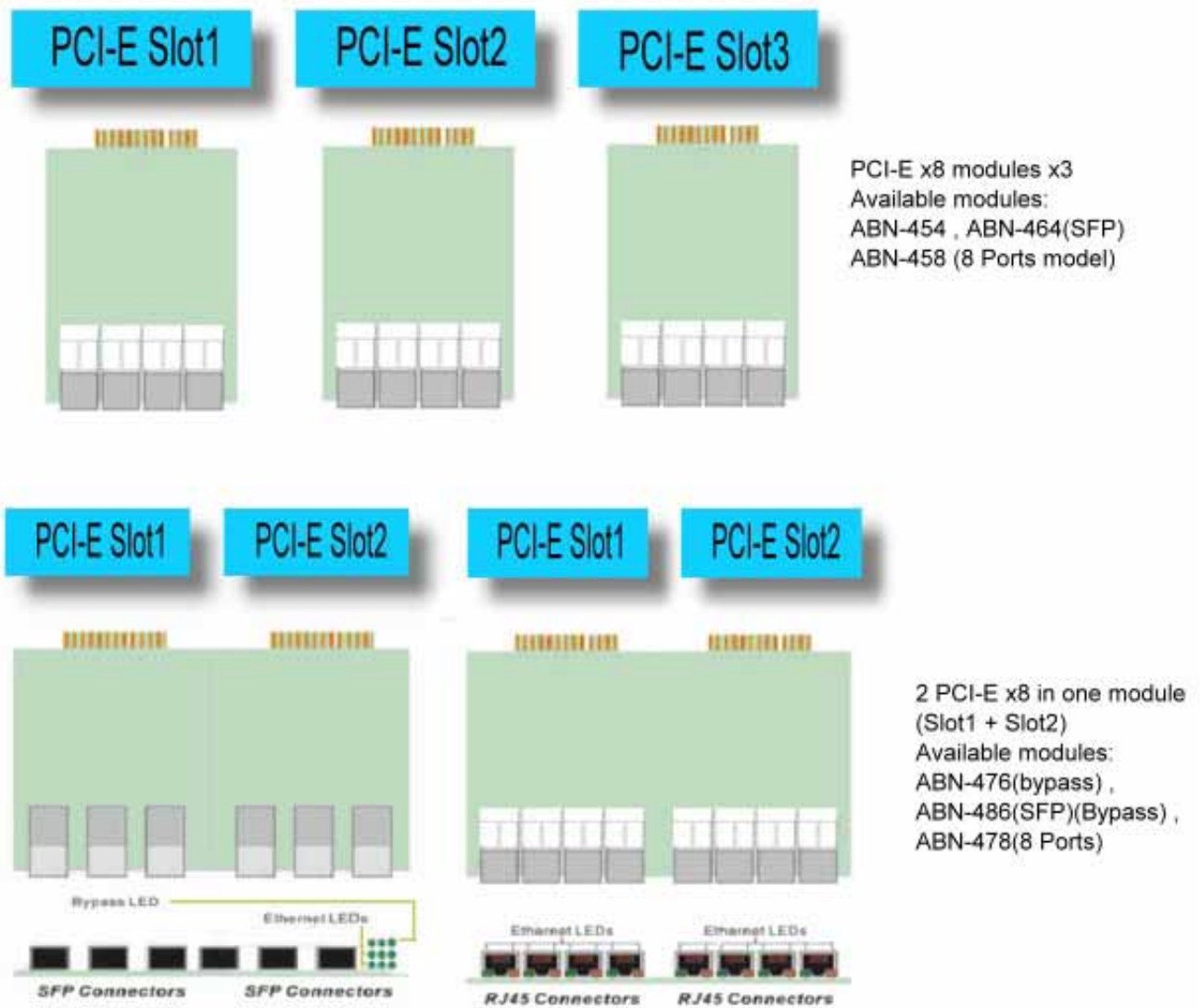


Figure 1-3 NAR-7090 Ethernet Module


1.6 Product Specifications

#	Requirement	Detailed Description
1.Processor		
1.1	Supplier	Intel®
1.2	CPU Model	Intel® Xeon® 5100 series CPU support list refer to Appendix A
1.3	Socket Type	LGA 771
1.4	Front Side Bus	1066/1333MHz
1.5	Chipset support	Intel 5000P (MCH) + 6321 ESB2 (I/O CH)
2.Memory		
2.1	Memory Type	ECC/Register FB DIMM 533/667MHz
2.2	Memory Slot	240pin FBDIMM * 8
2.3	Maximum	Maximum support up to 32GB
2.4	Channel	DIB and Dual channel
3.Expansion Slot		
3.1	64-bit PCI-X	Support 1* 64-bit 133/100MHz PCI-X expansion slots Support 2* 64-bit 133/100MHz PCI-X expansion slots (Option)
3.2	PCI-Express	PCI-Express X 1 ♦ Not Provide
		PCI-Express X 4 Support 1* PCI-E X4 interface from ESB2 on board
		PCI-Express X 8 Support 3* PCI-E x8 interface in proprietary interface
3.3	PICMG Interface	♦ Not Provide
4.I/O port		
4.1	IDE channels	Provide 1* 40 pin IDE connector
4.2	SATA channels	Provide 4 SATA connectors on board. Support 3Gbps SATA standard
4.3	CF slot	Provide 1 CF slot on board
4.4	USB	Provide 2 USB port on front panel <ul style="list-style-type: none"> Support USB 2.0. USB shall support for FDD, KBD, MOUSE, CD-ROM. CD-ROM and FDD shall be bootable devices
5. I/O Port		
5.1	EIA/TIA232	COM-1 connected as an external accessible RJ-45 connector. Pin-definition refers to Appendix-A. System ship with console cable.
6 Ethernet		
6.1	Gigabit Ethernet	Standard model will support <ul style="list-style-type: none"> 1>. 8 Gigabit Ethernet copper 2>. 8 Gigabit Ethernet ports with 4 copper and 4 SFP connector
6.2	10G Ethernet	Not provide on board, but with module support
6.3	Management	Provide 2* Gigabit Ethernet port as management function
6.4	Ethernet Function	Bypass Bypass function as model selection
		Dual Personality Dual Personality as Module option selection
6.5	Direction	Ethernet interface sequence in Linux OS should be in ascending order from left hand sites.
7. Video		
7.1	EZIO-3	Provide 2*16 character LCD display
7.2	LED	1 * Power LED (Green) – in front
8. Power supply		
8.1	Output Power	Provide 2 kind of power supply with active PFC control <ul style="list-style-type: none"> 1>. 400~460W redundant power supply 2>. 400~460W single power supply
8.2	Power on/off operation	<ul style="list-style-type: none"> There should be a on/off switch on PSU itself or a separated on/off switch attached to PSU to turn the PSU on/off; this switch is for AT mode operation There should be a separated toggle switch which is for ATX mode operation There is an “always on” item in the BIOS. System will be powered up automatically while power is resumed, if it is “on” before power failed ACPI signal in Linux will be generated while Toggle switch being pressed Customer can use a script file to specify the power off procedure
8.3	I2C function	Provide I2C function for system monitor P/S status
9. System Function		
9.1	H/W Reset and Load Factory	Provide a reset button in system panel Provide a pin-header on board connect to additional push button for F/D

#	Requirement	Detailed Description								
	Default buttons	(Load Factory Default). <ul style="list-style-type: none">The F/D can be connected to a toggle switch and the status of button pressing will be latched by h/w. The button pressing status can be read and cleared by s/w command.								
9.2	GPIO function	<ul style="list-style-type: none">								
9.3	Fan connector	Provide easy accessible Hot-Swap fan.								
9.4	Flash ROM	8 M bits								
9.5	Non-volatile Memory	<ul style="list-style-type: none">One non-volatile memory is needed via I2C channel.8KB memory should be a standard size and it is upgradeable to 64KB.								
9.6	System Monitoring Feature	Provide system monitor function that can monitor CPU temperature System Temperature and operating voltage.								
9.7	Watch Dog Timer	There should be two choices, reset or interrupt subroutine call, while WDT time-out.								
9.8	IPMI function	Provide IPMI 2.0 on socket								
10. System dimension										
10.1	Dimension	2U W: 430mm x D: 510mm x H: 88mm								
10.2	Front Panel	<ul style="list-style-type: none">EZIOUSB interface: dual-USB_connectorsRS232 interface: RS232 port with RJ45 connector for system console, tab-down, no LED.Hardware Reset ButtonPCI-X slots opening for optionEthernet interfaces:LED:<ul style="list-style-type: none">System LED: Power,Ethernet LED: For every Ethernet interface there should be LEDs for link status and speed of LAN-ports, which should be built in the connector. <u>For detailed signaling, please refer to Appendix-D LED Signaling.</u>								
10.3	Rear Panel	<ul style="list-style-type: none">AC power inletHot-swapped system Fan modules.								
10.4	Chassis Color	Standard Pantone Black-C								
10.5	Environmental requirements	<table><tr><td></td><td></td><td>Temperature</td><td>Humidity</td></tr><tr><td>Operating</td><td></td><td>5°C to 40°C.</td><td>10~90% RH</td></tr></table>			Temperature	Humidity	Operating		5°C to 40°C.	10~90% RH
		Temperature	Humidity							
Operating		5°C to 40°C.	10~90% RH							

1.7 LED Signaling Standard

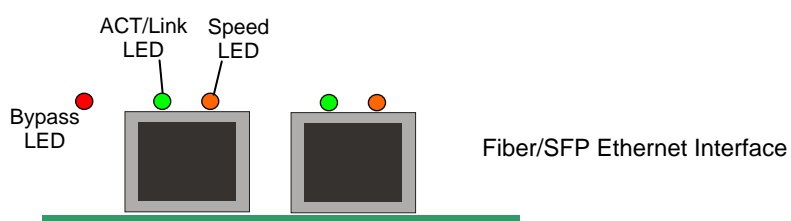
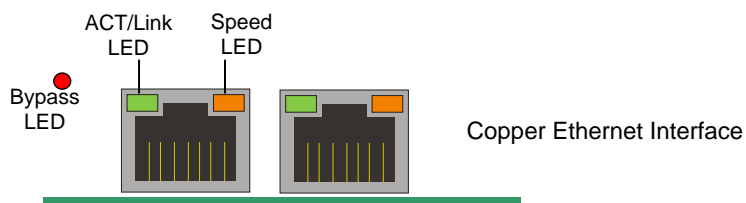
1. Power and Data-access LED

Lettering	Symbol	Function	Color	Signaling
PWR		Power status	Green	Off – No power, system off. On – Power good, system on.
HD LED	Power LED	HD Status	Green	Off – No power, No HD On – HD good, Have HD.
	Data Access	Data Access	Blue	Off – no data access through IDE or SATA channel On – data is in transition through IDE or SATA channel

2. Ethernet LED

Label	Color	Indication	Status
ACT/LINK	Green Or Others	On	1. The Ethernet port is receiving power. 2. Good linkage between the Ethernet port and its supporting hub.
		Off	1. The adapter and switch are not receiving power. 2. No connection between both ends of network cable. 3. The drivers of Ethernet have not been loaded or does not function correctly.

	Green Or Others	Flashing	The adapter is sending or receiving network data. The frequency of the flashes varies with the amount of network traffic.
SPEED	Yellow	On	ACT/LNK LED must on then this LED show the operating at 1000 Mbps. If ACT/LINK is off and this function will be disable.
	Green	On	ACT/LNK LED must on then this LED show the operating at 100 Mbps. If ACT/LINK is off and this function will be disable.
		Off	ACT/LNK LED must on then this LED show the operating at 10 Mbps. If ACT/LINK is off and this function will be disable.



3. Bypass LED

LED Status	green	red	off
Bypass Mode/Status	normal mode	bypass mode, triggered by WDT expiring	power off, in normal or bypass mode which is defined by customer

Chapter 2 Getting Started

This section describes how the hardware installation and system settings should be done.

2.1 Included Hardware

The following hardware is included in package:

- ◆ NAR-7090 Communication Appliance System Board
- ◆ One null serial port cable

2.2 Before You Begin

To prevent damage to any system board, it is important to handle it with care. The following measures are generally sufficient to protect your equipment from static electricity discharge:

When handling the board, to use a grounded wrist strap designed for static discharge elimination and touch a grounded metal object before removing the board from the antistatic bag. Handle the board by its edges only; do not touch its components, peripheral chips, memory modules or gold contacts.

When handling processor chips or memory modules, avoid touching their pins or gold edge fingers. Restore the communications appliance system board and peripherals back into the antistatic bag when they are not in use or not installed in the chassis.

Some circuitry on the system board can continue operating even though the power is switched off. Under no circumstances should the Lithium battery cell used to power the real-time clock be allowed to be shorted. The battery cell may heat up under these conditions and present a burn hazard.

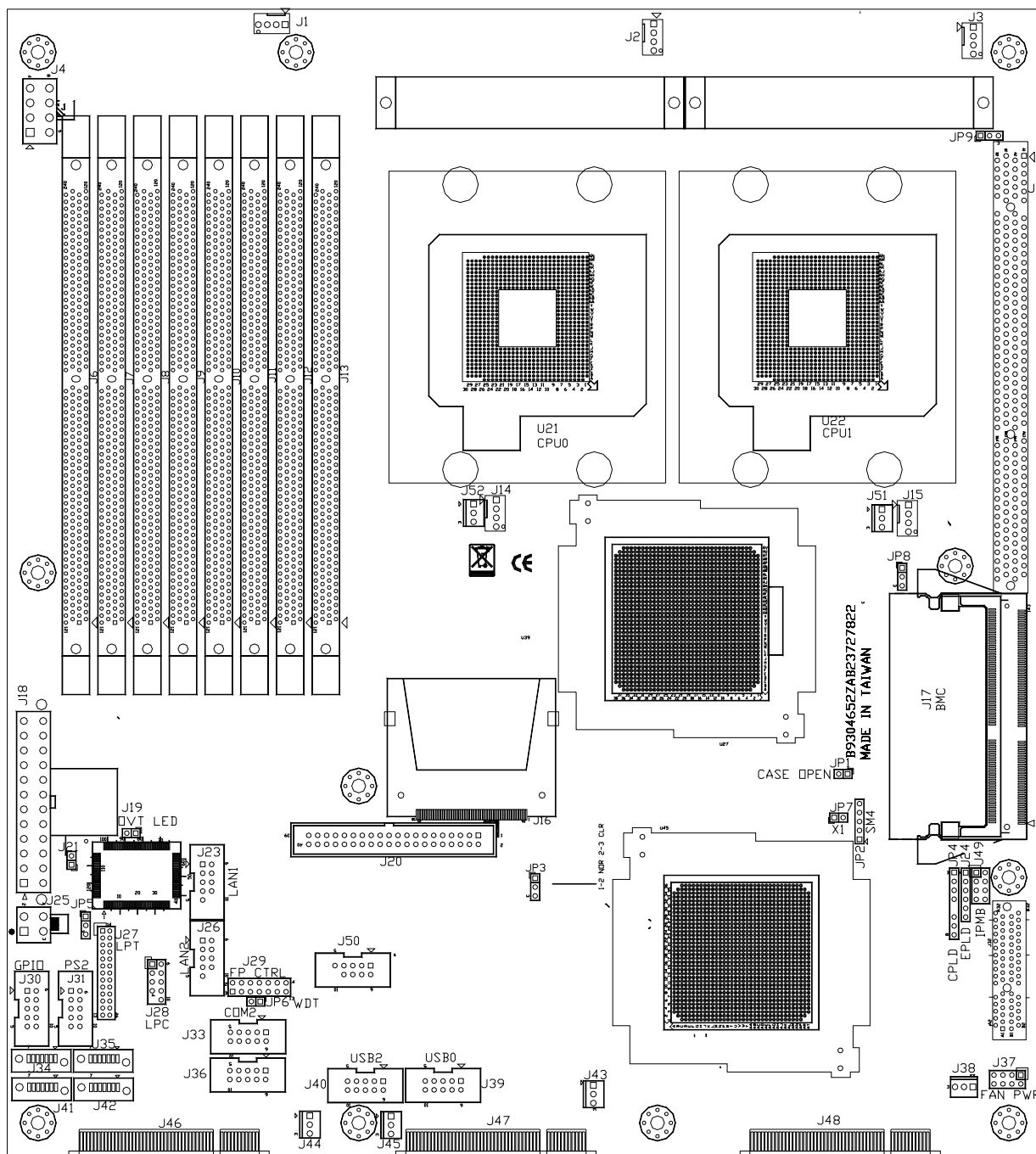
WARNING!

1. **"CAUTION: DANGER OF EXPLOSION IF BATTERY IS INCORRECTLY REPLACED. REPLACE ONLY WITH SAME OR EQUIVALENT TYPE RECOMMENDED BY THE MANUFACTURER. DISCARD USED BATTERIES ACCORDING TO THE MANUFACTURER'S INSTRUCTIONS"**
2. **This guide is for technically qualified personnel who have experience installing and configuring system boards. Disconnect the system board power supply from its power source before you connect/disconnect cables or install/remove any system board components. Failure to do this can result in personnel injury or equipment damage.**
3. **Avoid short-circuiting the lithium battery; this can cause it to superheat and cause burns if touched.**
4. **Do not operate the processor without a thermal solution. Damage to the processor can occur in seconds.**
5. **Do not block air vents. Minimum 1/2-inch clearance required.**

2.3.1 NAR-7090 System Board Jumper

In general, jumpers on NAR-7090 system board are used to select options for certain features. Some of the jumpers are configurable for system enhancement. The others are for testing purpose only and should not be altered. To select any option, cover the jumper cap over (Short) or remove (NC) it from the jumper pins according to the following instructions. Here NC stands for “Not Connected”.

Location of Jumpers



PPAP-3727L ZR2

JP1: CASEOPEN 2	JP2: SM BUS 4
JP3: RTC clear, 1-2:Normal 2-3: Clear	JP4: LC4128V CPLD download
JP5: GPIO power, 1-2:5V 2-3:3.3V	JP6: Watchdog Enable, in: Enable out: Disable
JP7: J32 PCI express (SB 0), out: x4 in: x1	JP8: 2-3 FSB1333(default), open FSB1066
JP9 1-2:J1~J3 4pin fan, 2-3: J1~J3 3pin fan(default 2-3)	
J1: Chassis Fan 3	J2: Chassis Fan 2
J3: Chassis Fan 3	J4: ATX PSU AUX +12V input

J5: PCI-x 64bit expansion slot	J6: FBD Channel 3 DIMM 1
J7: FBD Channel 3 DIMM 0	J8: FBD Channel 2 DIMM 1
J9: FBD Channel 2 DIMM 0	J10: FBD Channel 1 DIMM 1
J11: FBD Channel 1 DIMM 0	J12: FBD Channel 0 DIMM 1
J13: FBD Channel 0 DIMM 0	J14/J52: CPU 0 (left) FAN
J15/J51: CPU 1(right) FAN	J16: CF socket
J17: BMC socket	J18: ATX PSU main connector
J19: CASEOPEN 1	J20: IDE connector
J21: Over Temperature LED connector	J22: IPMB connector
J23: LAN 1	J24: XC9536XL_C CPLD download
J25: ATX PSU FAN power connector	J26: LAN 2
J27: Printer Port Connector	J28: LPC debug port
J29: Front Panel Control	J30: GPIO
J31: PS2 Keyboard / Mouse	J32: PCI express x4 expansion slot (SB 0)
J33: COM 2	J34: SATA 0
J35: SATA2	J36: COM 1
J37: FAN power option	J38: Front Fan
J39: USB 0/1	J40: USB 2/3
J41: SATA 0	J42: SATA 1
J43: Front Fan	J44: Front Fan
J45: Front Fan	J46: PCI express x8 expansion slot (NB 4/5)
J47: PCI express x8 expansion slot (NB 6/7)	J48:: PCI express x8 expansion slot (SB 1/2)
J49: IPMB	J50: LAN LED

J29: Front Panel Control

2 PWR LED+	4 PWR LED-	6 PWR ON-	8 PWR ON+	10 LDF-	12 LAN LINK+	14 LAN LINK-
1 IDE LED+	3 IDE LED-	5 RESET-	7 RESET+	9 LDF+	11 LAN ACT+	13 LAN ACT-

J30: GPIO

6 IO10	7 IO11	8 IO12	9 IO13	10 POWER
1 IO17	2 IO16	3 IO15	4 IO14	5 GND

J31: PS/2 KEYBOARD MOUSE

6 KEY DATA	7 NP	8 GND	9 KEY VCC	10 KEY CLK
1 MOUSE DATA	2 NP	3 GND	4 MOUSE VCC	5 MOUSE CLK

J23, J26 LAN

2 D1- (RJ45 2)	4 D2- (RJ45 6)	6 D3- (RJ45 5)	8 D4- (RJ45 8)	10 -
1 D1+ (RJ45 1)	3 D2+ (RJ45 3)	5 D3+ (RJ45 4)	7 D4+ (RJ45 7)	9 GND

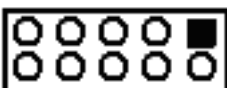
J50 LAN LED

2 1000- (LAN2,J26)	4 ACT- (LAN2)	6 LINK- (LAN2)	8 100- (LAN2)	10 -
1 1000- (LAN1,J23)	3 ACT- (LAN1)	5 LINK- (LAN1)	7 100- (LAN1)	9 GND

JP9 set to 2,3 Pin.

J39/J40: USB 0(1) / USB 2(3)

5 4 3 2 1



10 9 8 7 6

5 SBV0 (+5V)	4 SBD-1	3 SBD+1	2 GND	1 CHASSIS GND
10 CHASSIS GND	9 GND	8 SBD+0	7 SBD-0	6 SBV0 (+5V)

2.4 The Chassis

The system is integrated in a customized 2U chassis (**Fig. 2-1, Fig. 2-2**). On the front panel you will find a 4-push-button LCD module (EZIO), two USB ports and a COM port and Ethernet ports.



Fig. 2-1 Front view of the chassis



Fig. 2-2 Rear view of the chassis

2.5 Open the Chassis

1. Loosen the 2 screws of the chassis, three on each side and the rest two on the back, to remove the top lead (**Fig. 2-3**).



Fig. 2-3 Take off screws

2. The top lead (**Fig. 2-4**) can be removed from the base stand (**Fig. 2-5**).

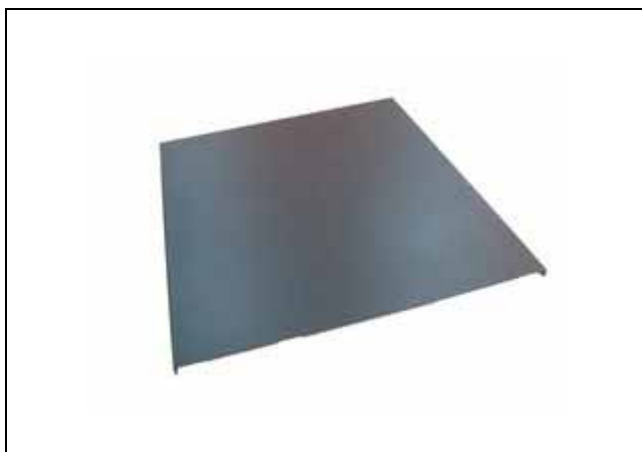


Fig. 2-4 The top lead



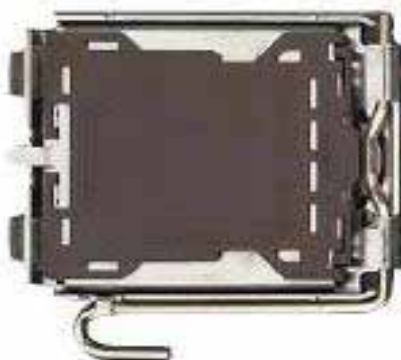
Fig. 2-5 The base stand

2.6 Install a Different Processor



To install a CPU

1. Local the CPU socket on the motherboard

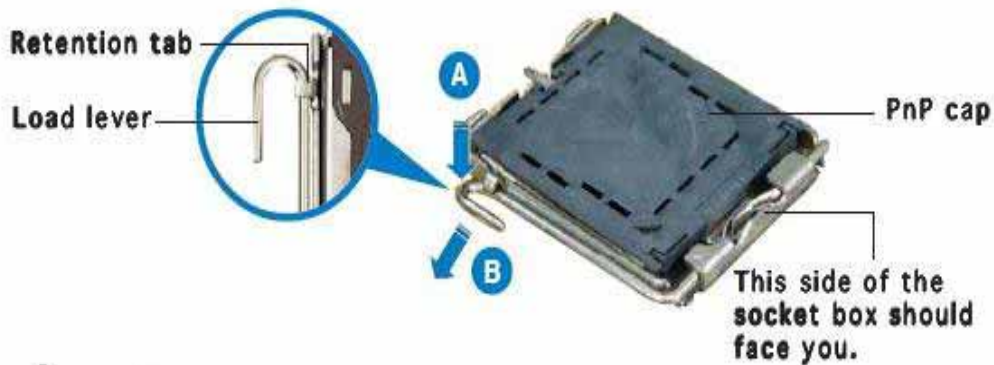


NAR-7090 CPU socket 771



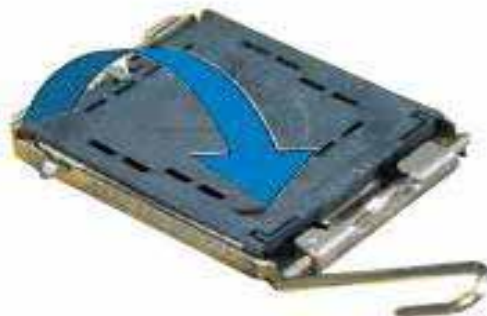
Before installing the CPU, make sure that the socket box is facing towards you and the load lever is on your left.

2. Press the load lever with your thumb (A), then move it to left (B) until it is released from the retention tab

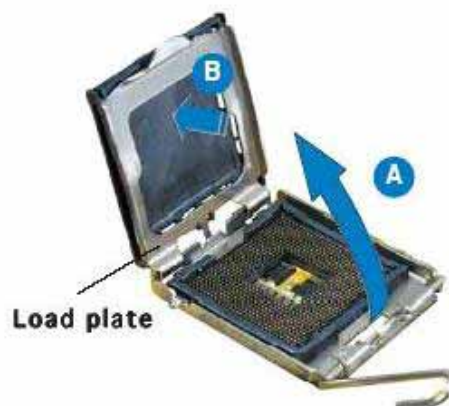


To prevent damage to the socket pins, do not remove the PnP cap unless you are installing a CPU.

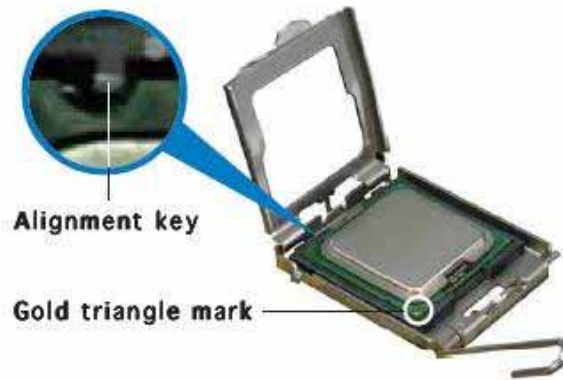
3. Lift the load lever in the direction of the arrow to a 135° angle



4. Lift the load plate with your thumb and forefinger to a 100° angle (A), then push the PnP cap from the load plate window to remove (B)



5. Position the CPU over the socket, making sure that the gold triangle is on the bottom-left corner of the socket. The socket alignment key should fit into the CPU notch



6. Close the load plate (A), then push the load lever (B) until it snaps into the retention tab



The CPU fits in only one correct orientation. DO NOT force the CPU into the socket to prevent bending the connectors on the socket and damaging the CPU!



Configure Processor Speed

The system was designed to self-detect its CPU speed. So it does not require any system adjustment.

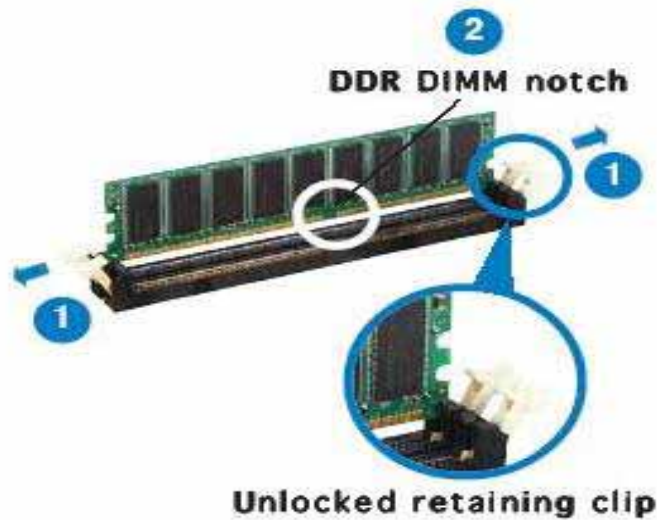
2.7 Remove and Install DIMM

Follow these steps to upgrade RAM module:



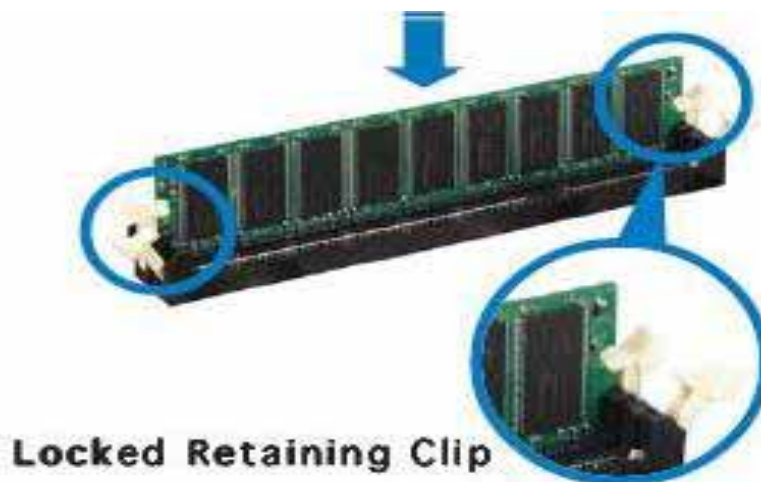
Make sure to unplug the power supply before adding or removing DIMMs or other system components. Failure to do so may cause severe damage to both the motherboard and the components.

1. Unlock a DIMM socket by pressing the retaining clips outward
2. Align a DIMM on the socket such that the notch on the DIMM matches the break on the socket



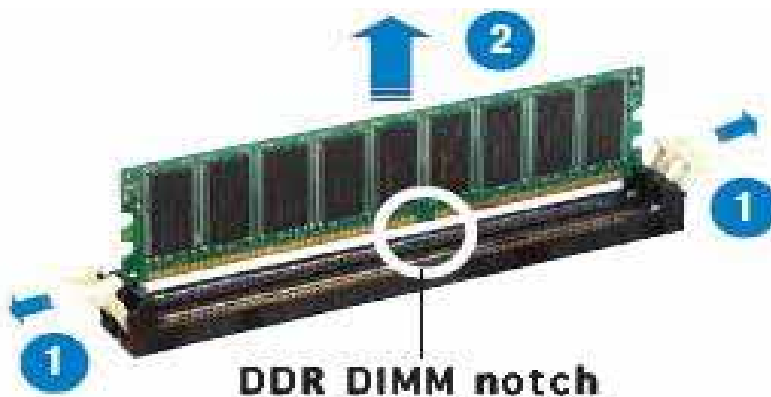
A DDR DIMM is keyed with a notch so that it fits in only one direction. DO NOT force a DIMM into a socket to avoid damaging the DIMM.

3. Firmly insert the DIMM into the socket until the retaining clips snap back in place and the DIMM is properly seated



Follow these steps to remove a DIMM:

1. Simultaneously press the retaining clips outward to unlock the DIMM



2. Remove the DIMM from the socket

2.8 Remove and Install Compact Flash Card

1. Insert the Compact Flash Card (**Fig. 2-7**) into the CF interface (**Fig. 2-8**).



Fig. 2-6 Compact Flash Card



Fig. 2-7 Insert Compact Flash Card into the CF interface

The completed installation of Compact Flash Card is shown as **Fig. 2-8**



Fig. 2-8 Completion of Compact Flash Card

2.9 Remove and Install Battery

1. Press the metal clip back to eject the button battery (**Fig. 2-9**).
2. Replace it with a new one by pressing the battery with fingertip to restore the battery (**Fig. 2-10**).

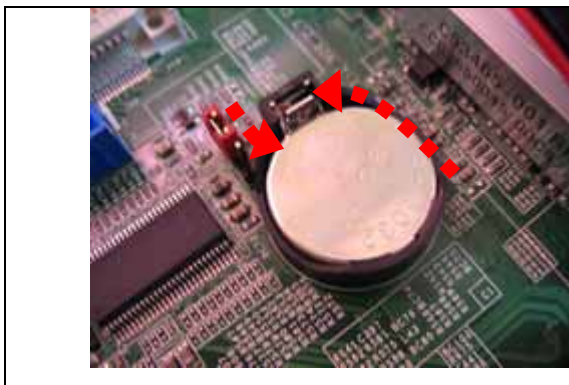


Fig. 2-9 Eject the battery



Fig. 2-10 Restore the battery

2.10 Install HDD

The system has an internal drive bay for one 3.5" SATA hard disk drive. If the HDD is not pre-installed, you can install it by yourself. Follow the steps below to install the HDD:

1. Fasten the four screws to lock HDD and bracket together (**Fig. 2-11a, 2-11b**).



Fig. 2-11a A 3.5" SATA HDD and the HDD bracket



Fig. 2-11b Fix HDD to the bracket

2. Connect Power cable and HDD cable to NAR-7090RB system board (**Fig. 2-12**).



Fig. 2-12a Connect HDD bracket to NAR-7090 system then push the switch in.



Fig. 2-12b Fix HDD into NAR-7090 system

2.11 Install Riser Card

The system has an internal riser card to support up to two PCI-X slots.

1. Fasten the six screws to lock riser and bracket together and fix in NAR-7090 (**Fig. 2-11a, 2-11b**).



Fig. 2-11a A ABR-162 riser card and the bracket



Fig. 2-11b Fix riser card to NAR-7090

2.12 Ear Mount Kit Installation

The NAR-7090 series shipped with 2 ear mount kits. The following is the installation instruction of these ear mounts:

1. Take out the L shape ear mount kits. One ear mount fits on one side of the chassis,
2. Placing the side with four holes against the chassis and the side with two holes face outward. (**Fig. 2-13**)
3. Fasten four screws on each side (**Fig. 2-13**)

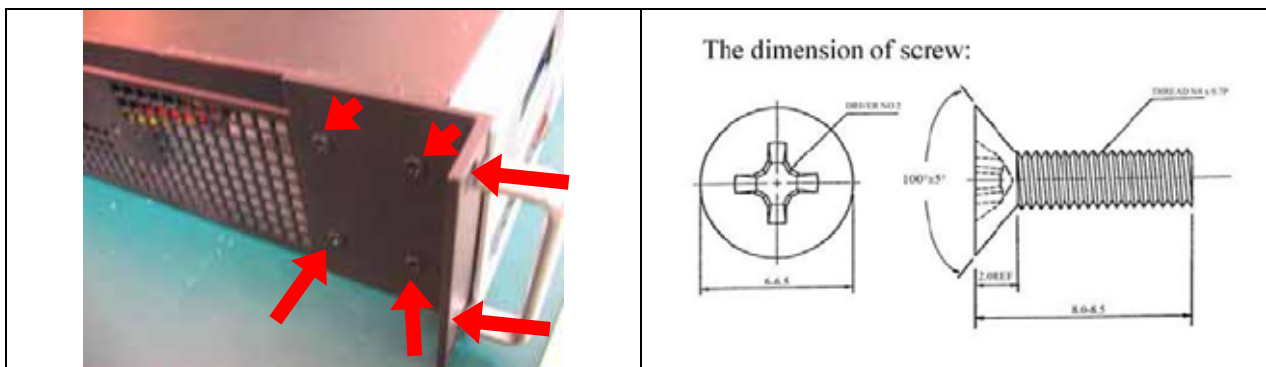


Fig.2-13 Fasten the screws to the side

2.13 Remove EZIO / LCD

The NAR-7090 series support EZIO modules. The following is the remove instruction of these EZIO/LCD modules:

1. Remove the system front panel first.
2. Remove all cables from EZIO

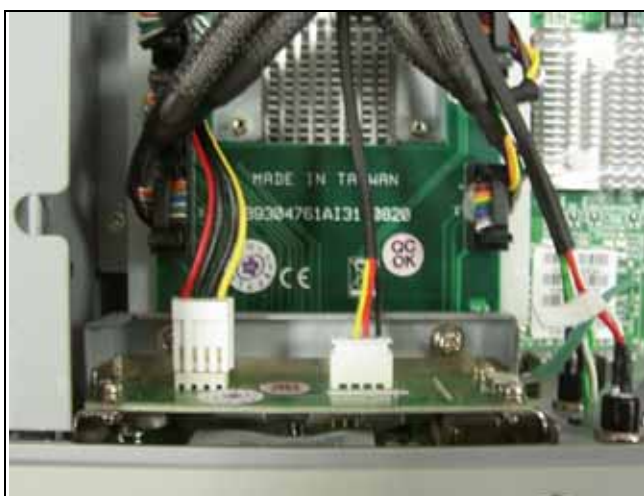


Fig.2-14 Remove the cable from EZIO



Fig.2-15 After remove the cable from EZIO

3. Remove the screws from chassis.



Fig.2-16 Remove the screws from EZIO

4. Remove the screws from EZIO Kit

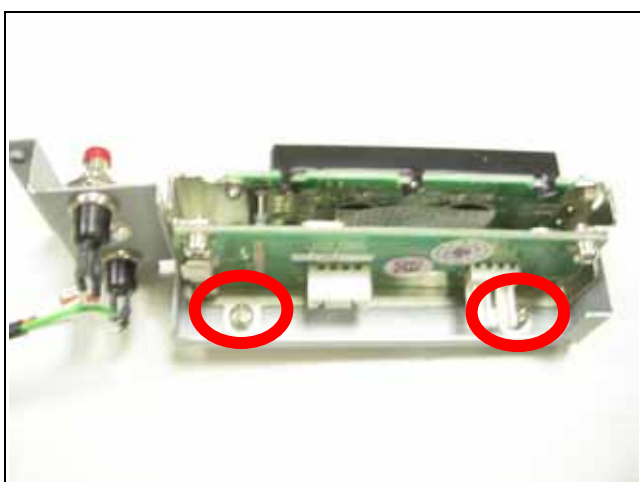


Fig.2-17 Remove the screws from EZIO kit

5. Final remove the EZIO/LCD module.



Fig.2-18 Finish

2.14 Remove Power Supply

The following is the remove step instruction of power supply.

1. Remove following screws from chassis first.

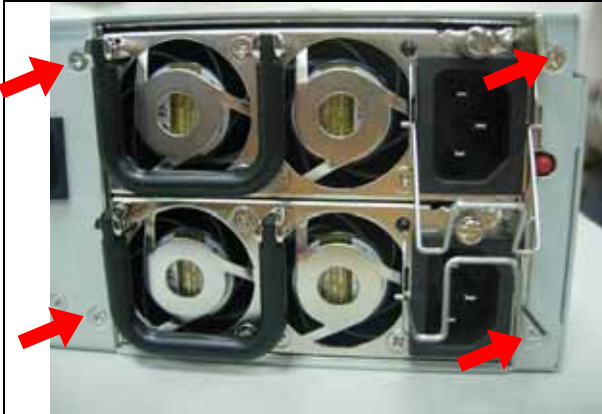


Fig.2-19 Remove the screws from rear chassis

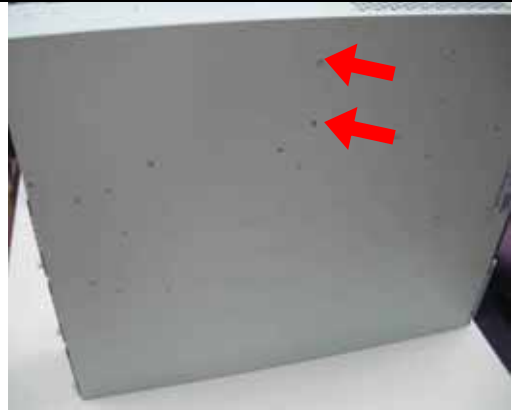


Fig.2-20 Remove screws from bottom chassis.

2. Remove all power cables from main board and HDD bay.

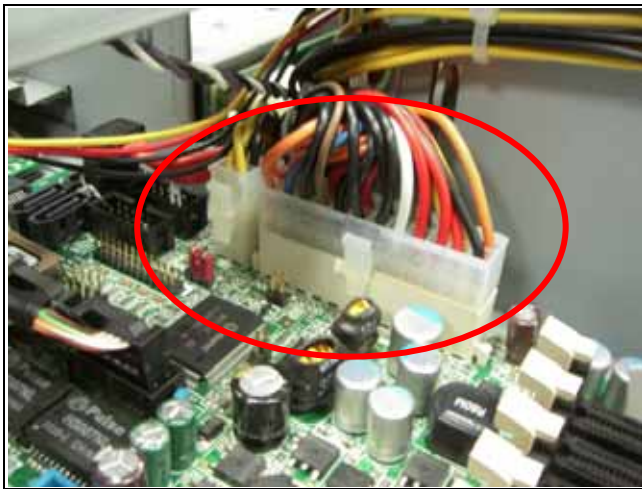


Fig.2-21 Remove power cables from board

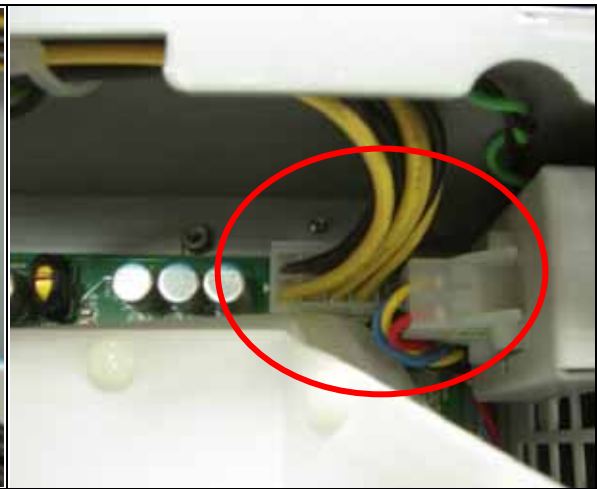


Fig.2-22 Remove power cables from board

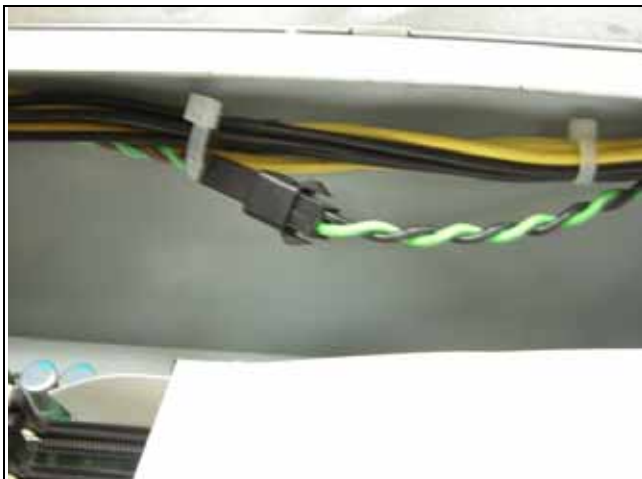


Fig.2-23 Remove power switch cables



Fig.2-24 Remove HD cables from HD Bay

3. Push the power supply inside system then lift up power supply to pull out the power supply.



Fig.2-25.1 Push the power supply inside system.



Fig.2-25.2 Pull out the power supply.

4. If user need to change power supply modules. Please follow following steps:



Fig.2-26.1 Loosen the screw to unlock the power module.

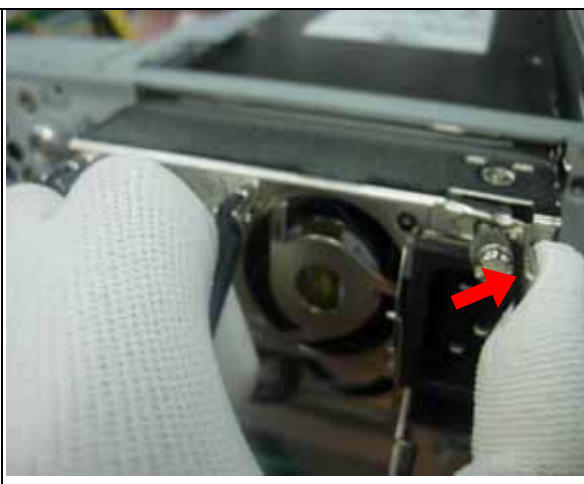


Fig.2-26.2 Pull out the power modules.

2.15 Remove main board

The following is the remove step instruction of main board.

1. Remove all add-on modules or riser card devices from system first



Fig.2-27 Remove Front bezel

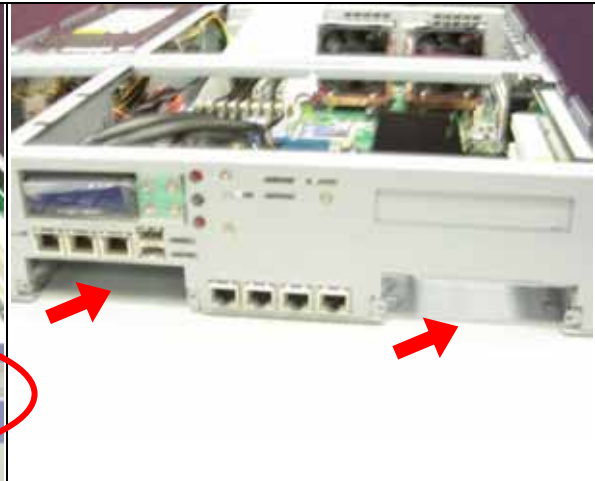


Fig.2-28 Remove all add-on modules.

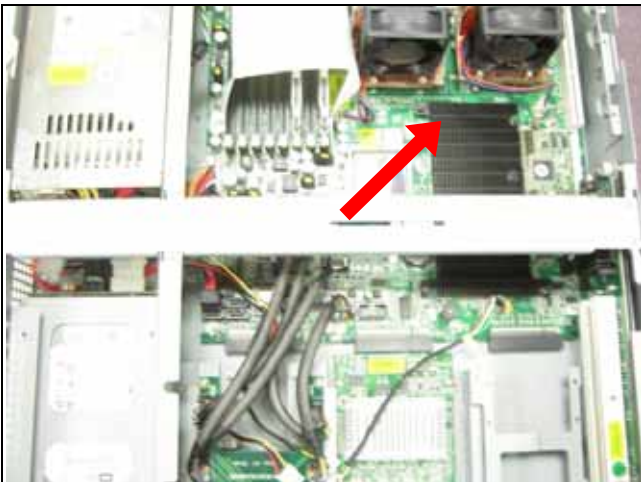


Fig.2-29 Remove middle bracket

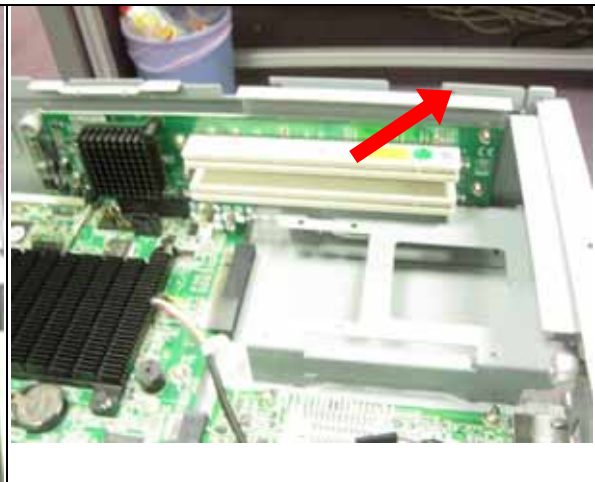


Fig.2-30 Remove riser card.



Fig.2-31 Remove Front bracket

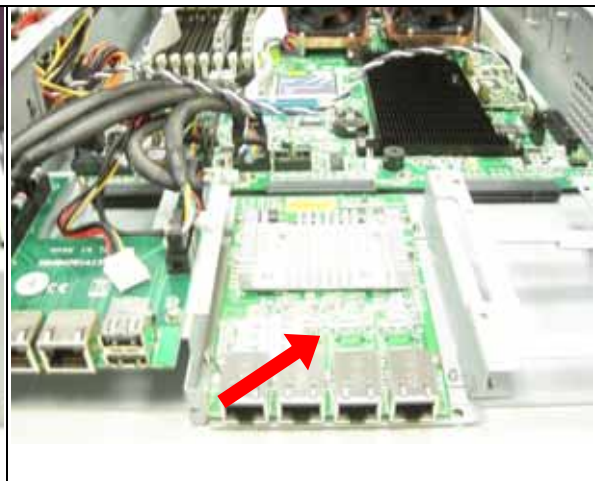


Fig.2-32 Remove slot B add-on card

2. Remove following items from main board: Cables, CPU cooler, CPU, memory ,CF ,IPMI , System Fan.

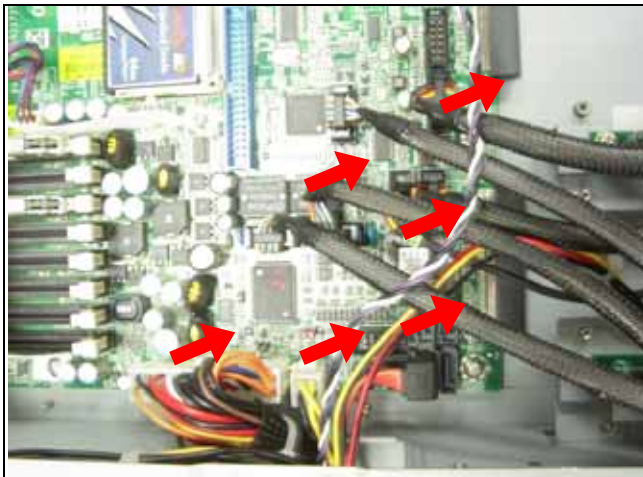


Fig.2-33 Remove all cables. from board.

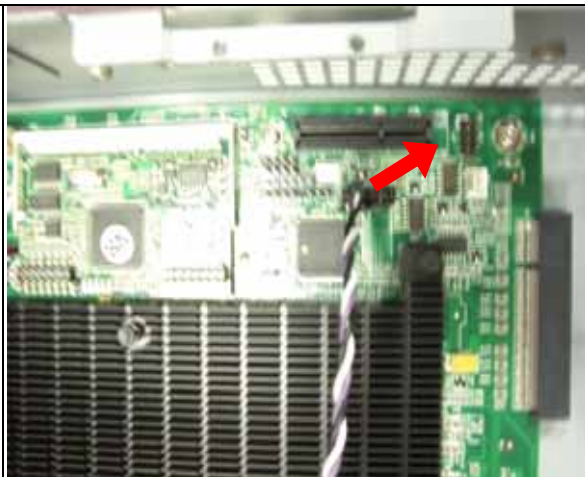


Fig.2-33.2 Remove all cable from board.

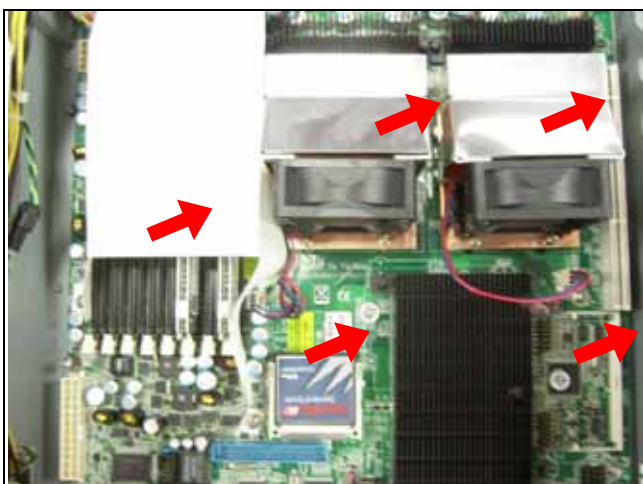


Fig.2-34 Remove CPU cooler, CPU , memory , CF and IPMI module.

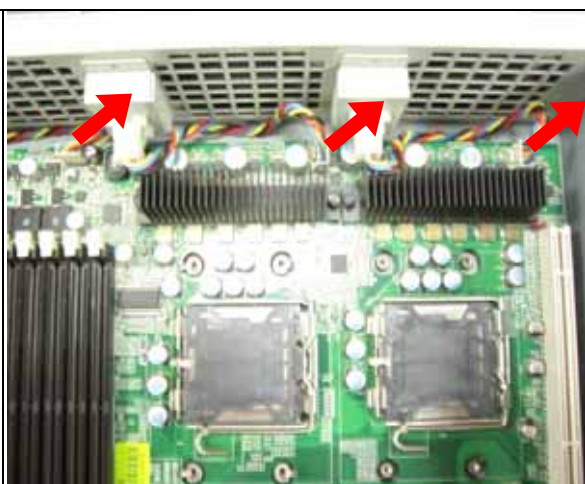


Fig.2-35 Remove System Fan

3. After remove above items, and push the PnP cap back to CPU socket. Users can start remove all screws from board.



Fig.2-36 Remove all screws from main board.

4. After remove all screws from board. User can remove main board. Please be gentle and carefully. Avoid colliding board with chassis bottom sticks. It may damage the main components.

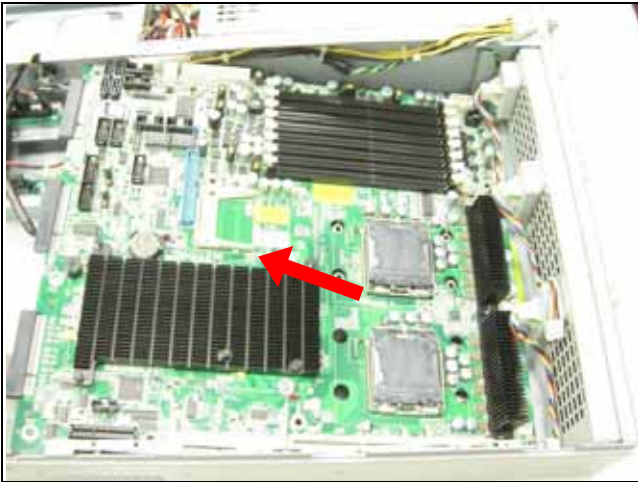


Fig.2-37 lift up main board from rear side of system

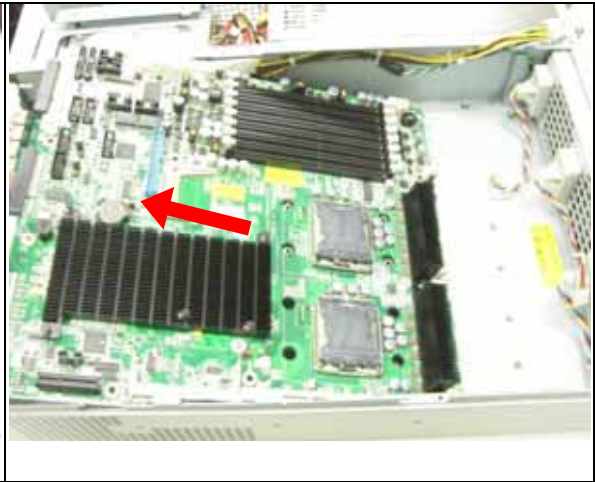


Fig.2-38 Lift the board up for moving.

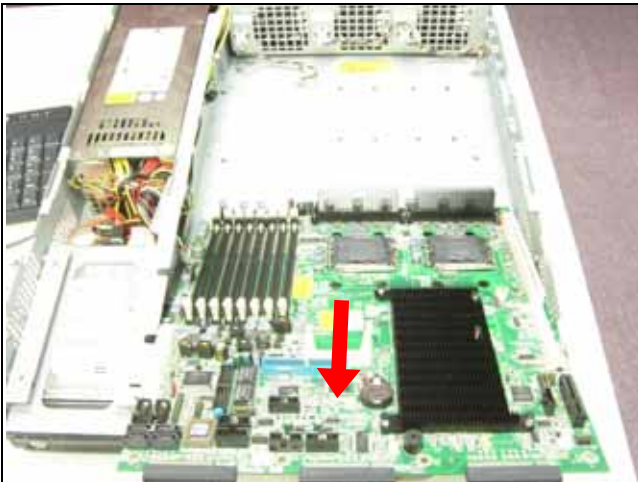


Fig.2-39 Remove main board from rear side

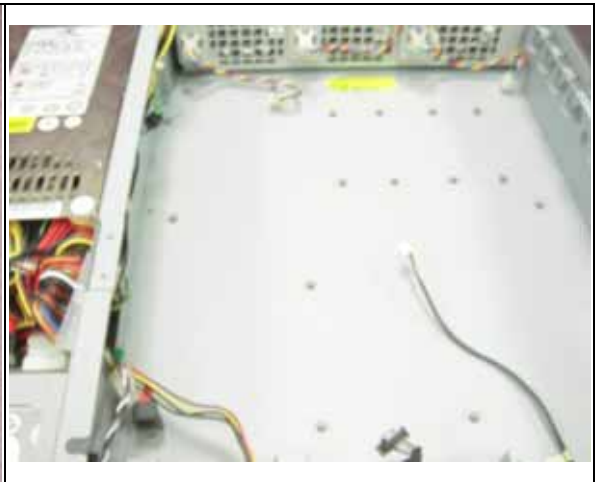


Fig.2-35 Finish

2.16 Use a Client Computer



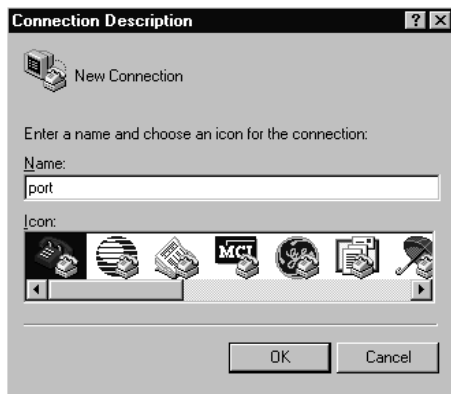
Connection Using Hyper Terminal

If users use a headless NAR-7090 system, which has no mouse/keyboard and VGA output connected to it, the console may be used to communicate with NAR-7090.

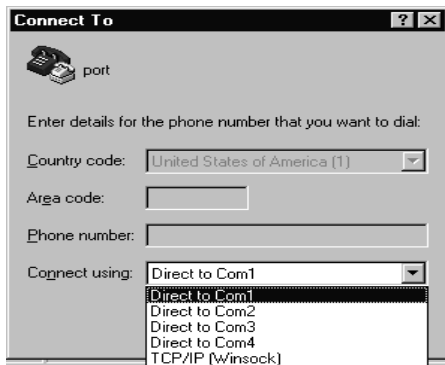
To access NAR-7090 via the console, Hyper Terminal is one of many choices. Follow the steps below for the setup:

Note: Terminal software may need to update for correct console output.

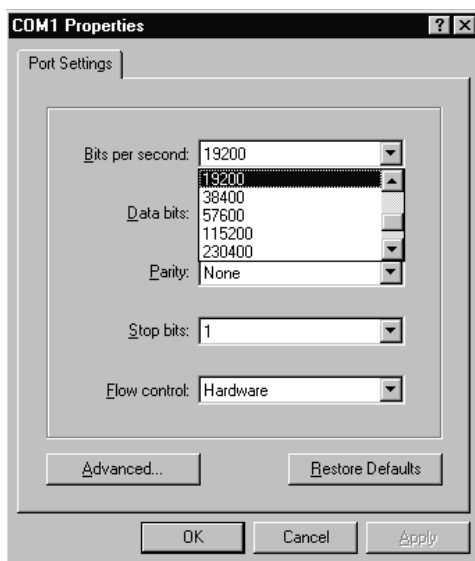
1. Execute HyperTerminal under C:\Program Files\Accessories\HyperTerminal
2. Enter a name to create new dial



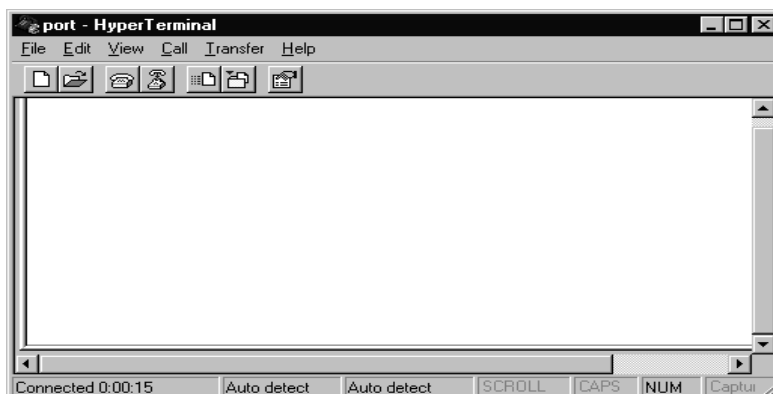
3. For the connection settings, make it Direct to Com1.



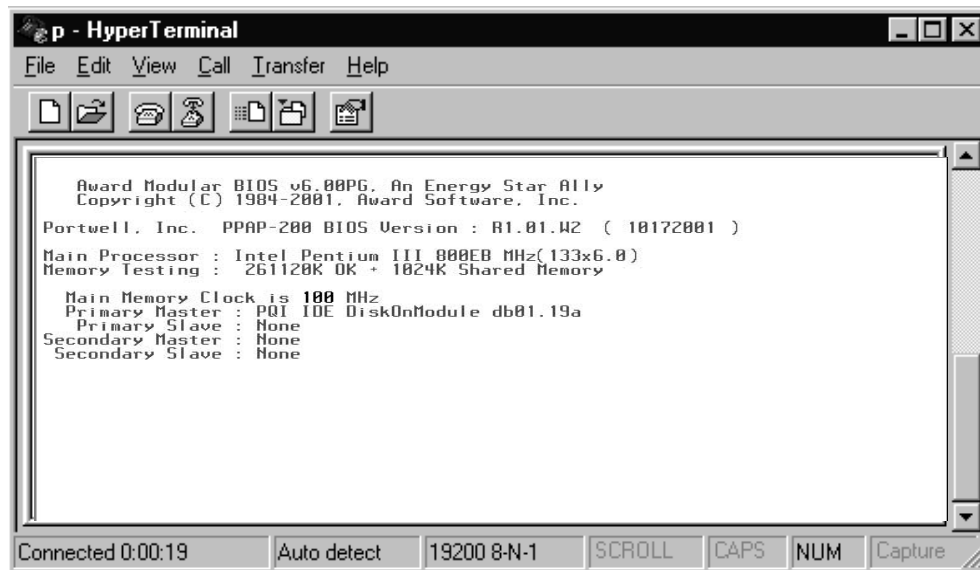
4. Please make the port settings to Baud rate 19200, Parity None, Data bits 8, Stop bits 1



5. Turn on the power of NAR-7090 system, after following screen was shown:



6. You can then see the boot up information of NAR-7090.



7. When message "Hit if you want to run Setup" appear during POST, after turning on or rebooting the computer, press **<Tab>** key *immediately* to enter BIOS setup program.

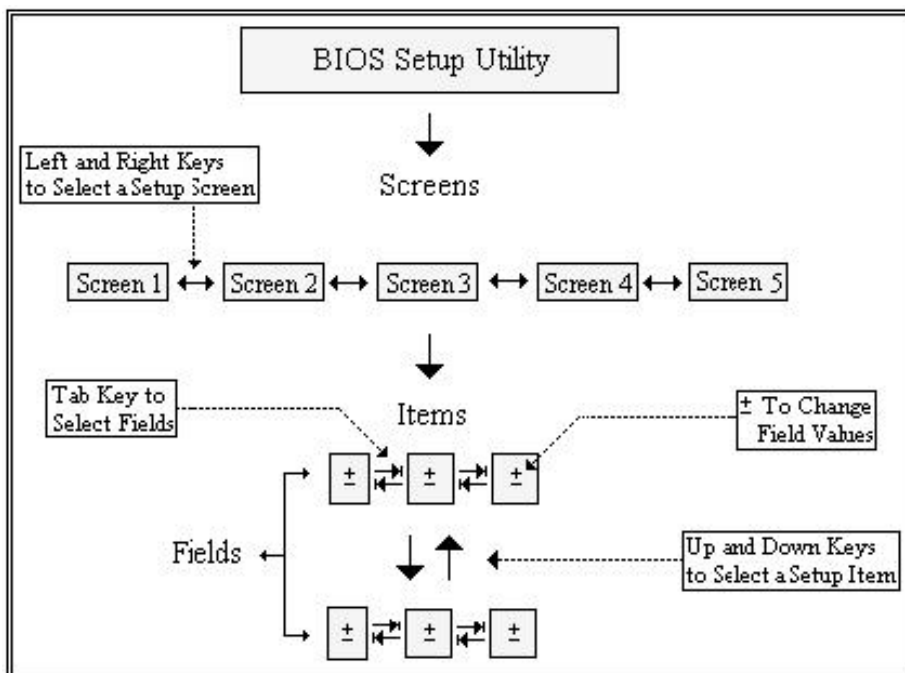
This is the end of this section. If the terminal did not port correctly, please check the previous steps.

Chapter 3 BIOS Setting

BIOS Setup Information

Power on the system, press the to run BIOS setup. After press the <Delete> key, the main BIOS setup menu displays. You can access the other setup screens from the main BIOS setup menu, such as the Chipset and Power menus.

The BIOS setup/utility uses a key-based navigation system called hot keys. Most of the BIOS setup utility hot keys can be used at any time during the setup navigation process. These keys include <F1>, <F10>, <Enter>, <ESC>, <Arrow> keys, and so on.



Control Keys

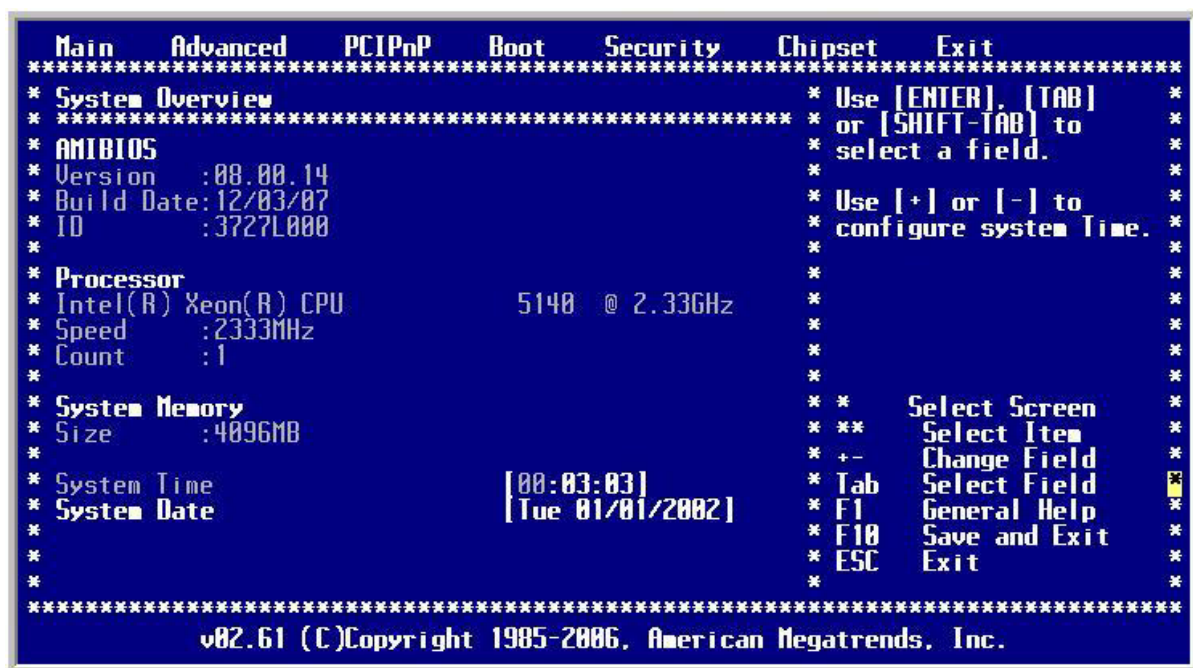
Key	Function
↑↓Up /Down	The <i>Up and Down</i> <Arrow> keys allow you to select a setup item or sub-screen.
→ ← Left/Right	The <i>Left and Right</i> <Arrow> keys allow you to select a setup screen. For example: Main screen, Advanced screen, Chipset screen, and so on.
+ - Plus/ Minus	The <i>Plus and Minus</i> <Arrow> keys allow you to change the field value of a particular setup item. For example: Date and Time.
Tab	The <Tab> key allows you to select setup fields.

Hot Key	Description
F1	<p>The <F1> key allows you to display the <i>General Help</i> screen.</p> <p>Press the <F1> key to open the <i>General Help</i> screen.</p> <div data-bbox="395 255 1358 680" data-label="Image"> </div>
F10	<p>The <F10> key allows you to save any changes you have made and exit Setup. Press the <F10> key to save your changes. The following screen will appear:</p> <div data-bbox="395 770 1358 960" data-label="Image"> </div> <p>Press the <Enter> key to save the configuration and exit. You can also use the <Arrow> key to select <i>Cancel</i> and then press the <Enter> key to abort this function and return to the previous screen.</p>
ESC	<p>The <Esc> key allows you to discard any changes you have made and exit the Setup. Press the <Esc> key to exit the setup without saving your changes. The following screen will appear:</p> <div data-bbox="395 1151 1358 1344" data-label="Image"> </div> <p>Press the <Enter> key to discard changes and exit. You can also use the <Arrow> key to select <i>Cancel</i> and then press the <Enter> key to abort this function and return to the previous screen.</p>
Enter	<p>The <Enter> key allows you to display or change the setup option listed for a particular setup item. The <Enter> key can also allow you to display the setup sub- screens.</p>



Main Menu

When you first enter the Setup Utility, you will enter the Main setup screen. You can always return to the Main setup screen by selecting the *Main* tab. There are two Main Setup options. They are described in this section.



System Date / Time

Use this option to change the system time and date. Highlight System Time or System Date using the <Arrow> keys. Enter new values through the keyboard. Press the <Tab> key or the <Arrow> keys to move between fields. The date must be entered in MM/DD/YY format. The time is entered in HH:MM:SS format.

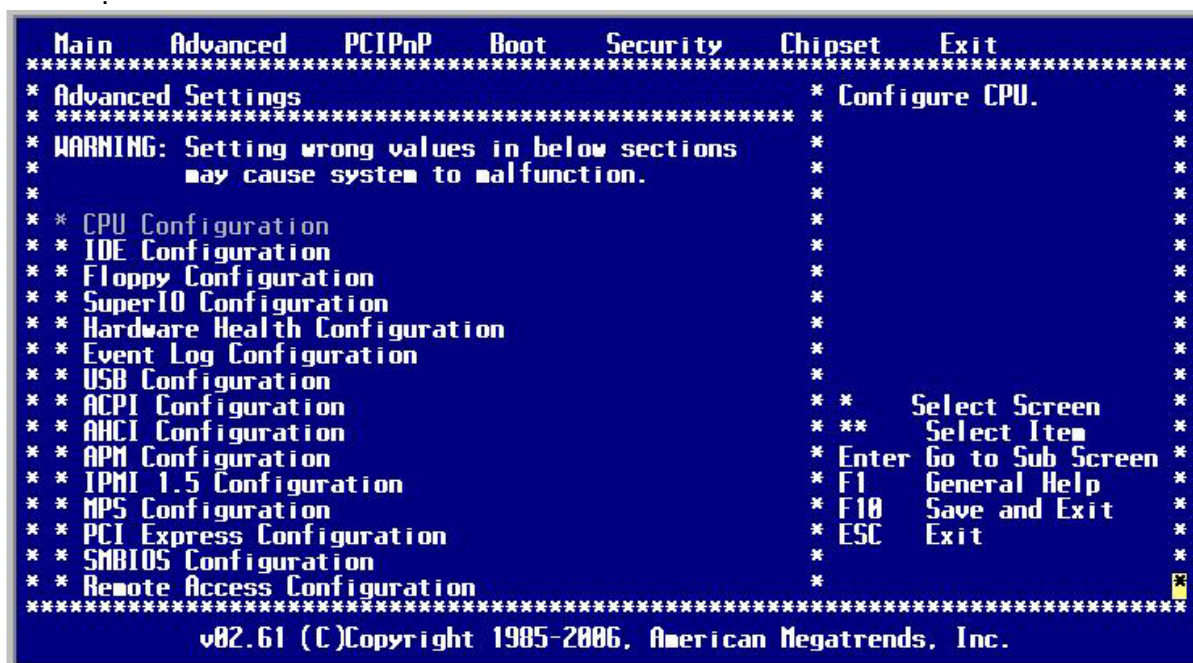
➤ Advanced BIOS Setup

Select the *Advanced* tab from the setup screen to enter the Advanced BIOS Setup screen.

Select any of the items in the left frame of the screen, such as SuperIO Configuration, to go to the sub menu for that item. It will display an Advanced BIOS

Setup option by highlighting it using the <Arrow> keys. All Advanced BIOS Setup options are described in this section. The Advanced BIOS Setup screen is shown below.

The sub menus are described on the following pages.



➤ IDE Configuration Setup

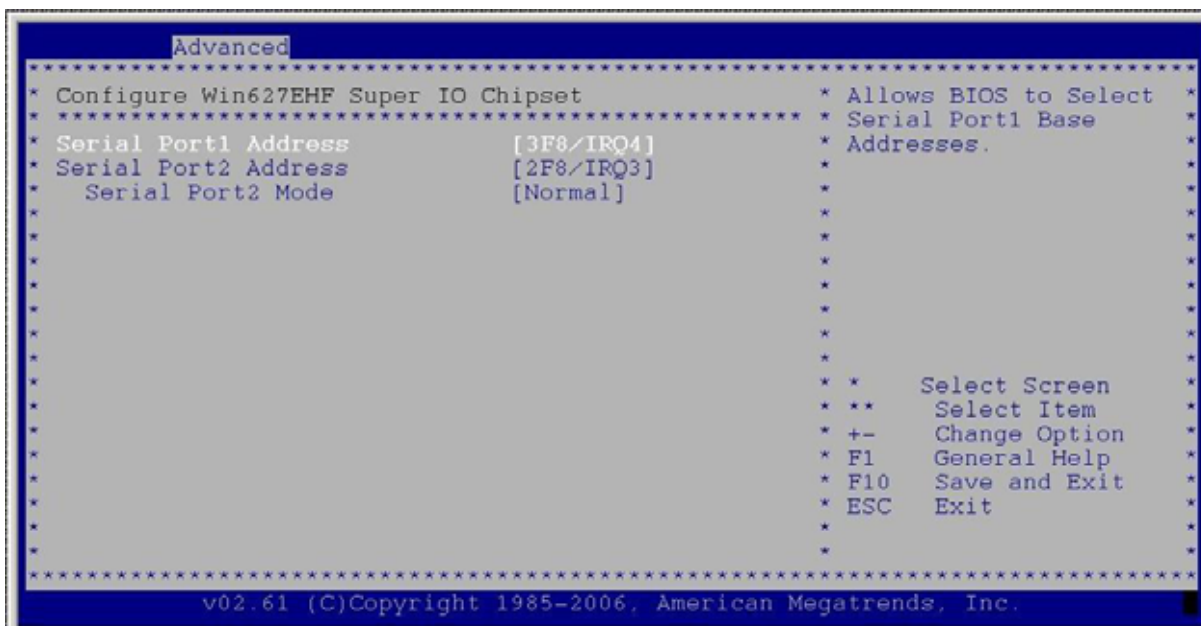
From the IDE Configuration screen, press <Enter> to access the sub menu. Use the up and down <Arrow> keys to select an item. The settings are described on the following pages.



➤ SUPER IO CONFIGURATION

SuperIO Configuration

Use this screen to select options for the Super I/O settings. Use the up and down <Arrow> keys to select an item. Use the <Plus> and <Minus> keys to change the value of the selected option. The settings are described on the following pages. The screen is shown below.



➤ REMOTE ACCESS CONFIGURATION

Remote Access Configuration

Use this screen to select options for the Remote Access Configuration. Use the up and down <Arrow> keys to select an item. Use the <Plus> and <Minus> keys to change the value of the selected option. The settings are described on the following pages. The screen is shown below.



Remote Access

Disable or enable the BIOS remote access feature here.

Serial Port Number

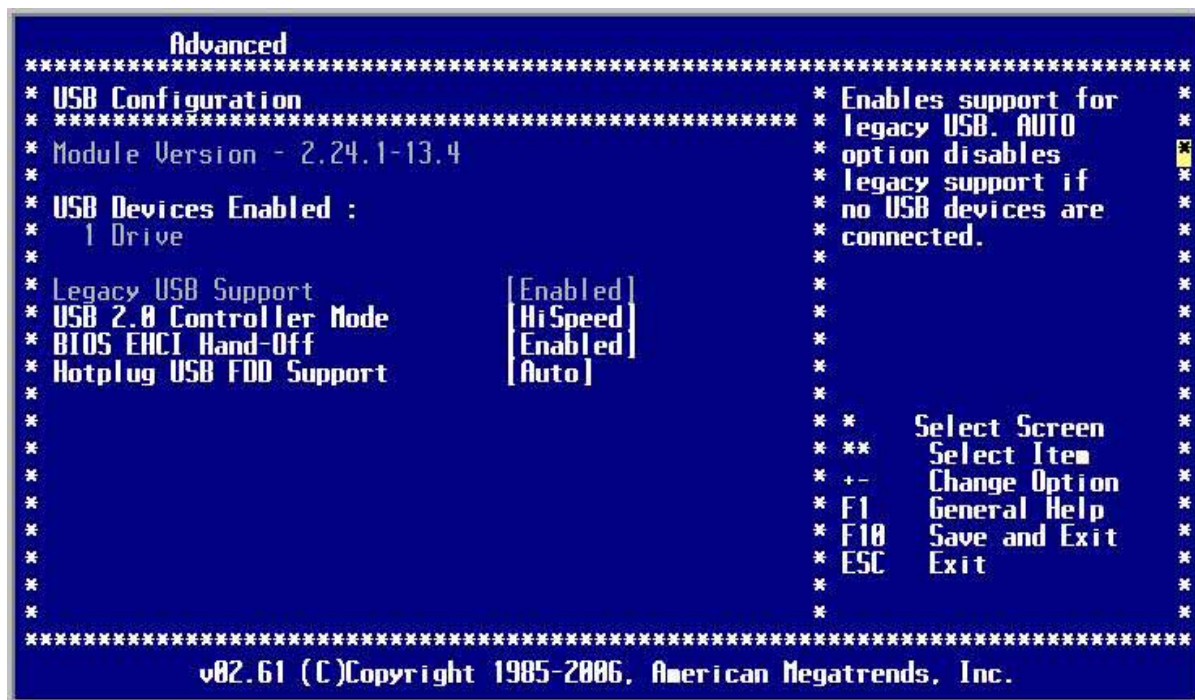
Select the serial port to use for console redirection. Set the value for this option to either COM1 or COM2.

Serial Port Mode

Select the baud rate you want the serial port to use for console redirection.

➤ USB Configuration

Use this screen to select options for the USB Configuration. Use the up and down <Arrow> keys to select an item. Use the <Plus> and <Minus> keys to change the value of the selected option. The settings are described on the following pages. The screen is shown below.



Legacy USB Support

Legacy USB Support refers to the USB mouse and USB keyboard support. Normally if this option is not enabled, any attached USB mouse or USB keyboard will not become available until a USB compatible operating system is fully booted with all USB drivers

loaded. When this option is enabled, any attached USB mouse or USB keyboard can control the system even when there is no USB drivers loaded on the system. Set this value to enable or disable the Legacy USB Support. The Optimal and Fail-Safe default setting is *Disabled*.

➤ CPU Configuration

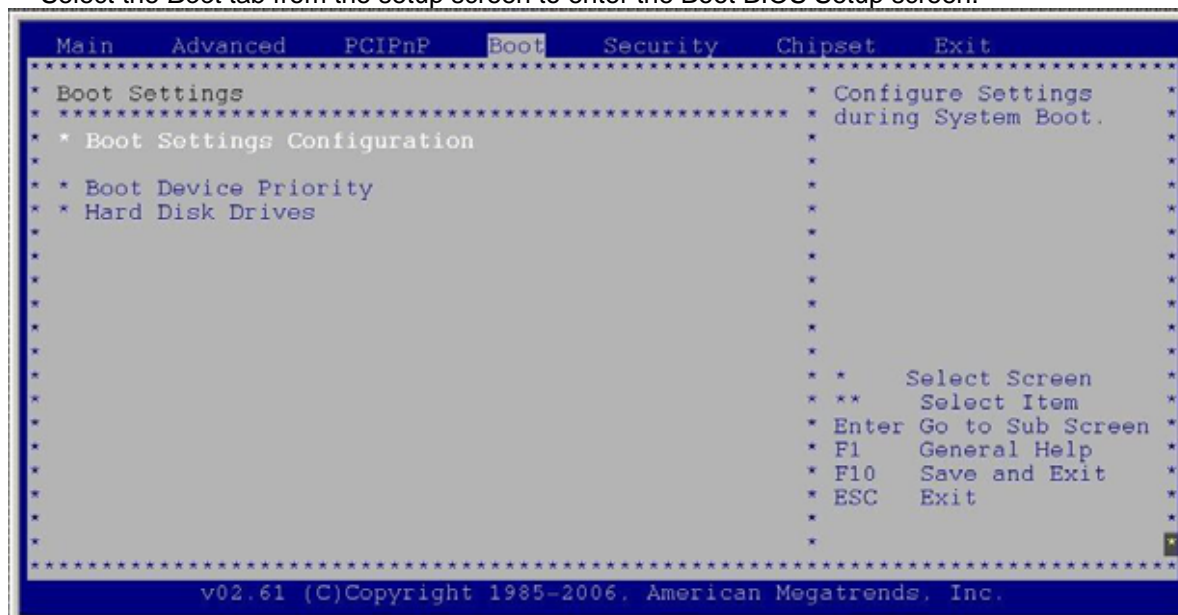
Use this screen to select options for the CPU Configuration. Use the up and down <Arrow> keys to select an item. Use the <Plus> and <Minus> keys to change the value of the selected option.



Note: The CPU Configuration setup screen varies depending on the installed processor.

➤ Boot Settings

Select the *Boot* tab from the setup screen to enter the Boot BIOS Setup screen.

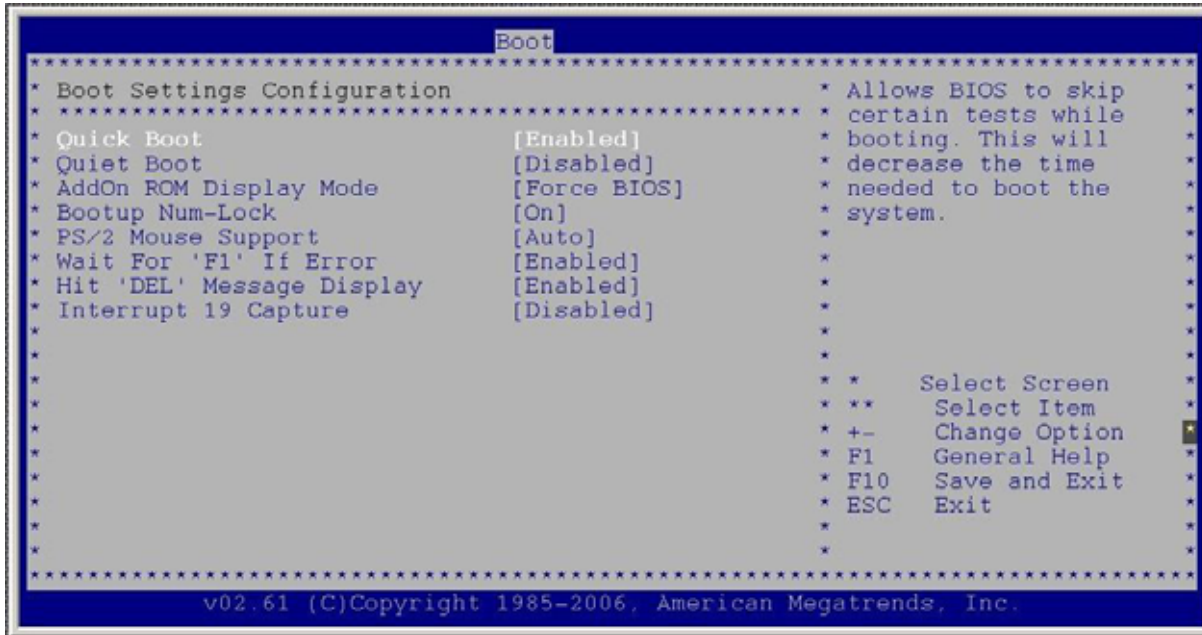


➤ BOOT SETTINGS CONFIGURATION SCREEN

Boot Settings Configuration

Use this screen to select options for the Boot Settings Configuration. Use the up and down <Arrow> keys to select an item. Use the <Plus> and <Minus> keys to change the value of the selected option. The settings are described on the following pages. The

screen is shown below.



Quick Boot

The Optimal and Fail-Safe default setting is *Disabled*.

Quiet Boot

Set this value to allow the boot up screen options to be modified between POST messages or OEM logo. The Optimal and Fail-Safe default setting is *Enabled*.

Add-On ROM Display Mode

Set this option to display add-on ROM (read-only memory) messages. The Optimal and Fail-Safe default setting is *Force BIOS*. An example of this is a SCSI BIOS or VGA BIOS.

Boot up Num-Lock

Set this value to allow the Number Lock setting to be modified during boot up. The Optimal and Fail-Safe default setting is *On*.

PS/2 Mouse Support

Set this value to allow the PS/2 mouse support to be adjusted. The Optimal and Fail-Safe default setting is *Enabled*.

Interrupt 19 Capture

Set this value to allow option ROMs such as network controllers to trap BIOS interrupt 19.

➤ BOOT DEVICE PRIORITY

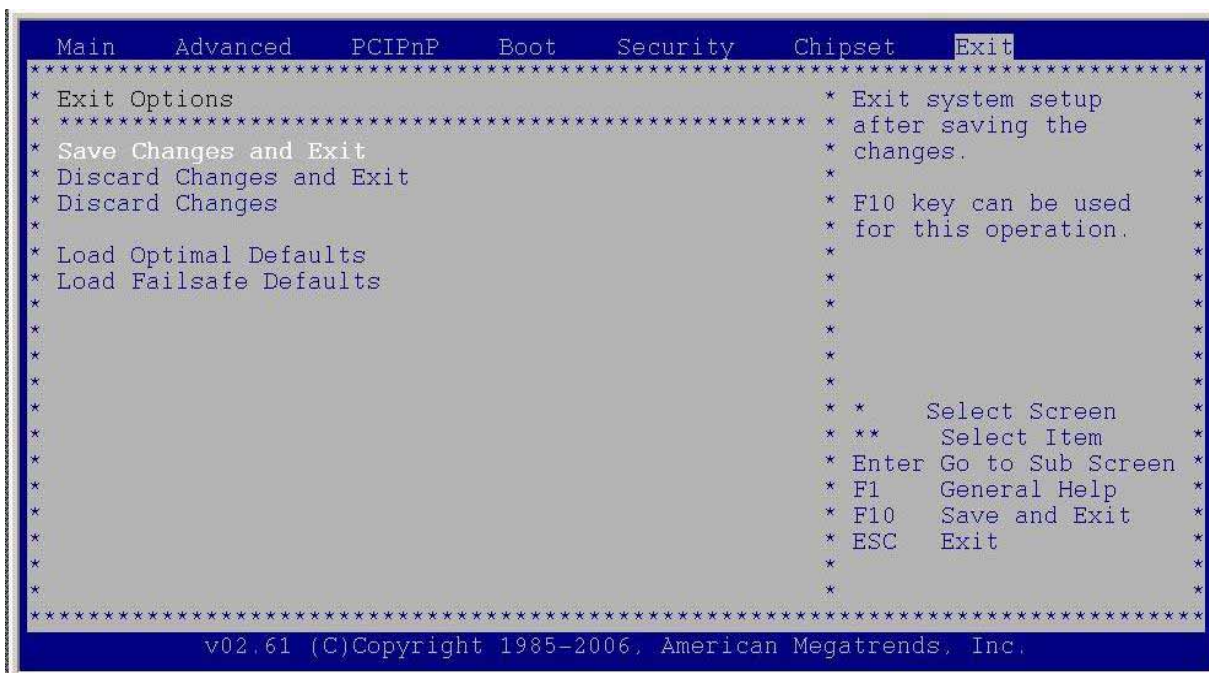
Boot Device Priority

Use this screen to specify the order in which the system checks for the device to boot from. To access this screen, select Boot Device Priority on the Boot Setup screen and press <Enter>. The following screen displays:



➤ Exit Menu

Select the *Exit* tab from the setup screen to enter the Exit BIOS Setup screen. Display an Exit BIOS Setup option by highlighting it using the <Arrow> keys. All Exit BIOS Setup options are described in this section. The Exit BIOS Setup screen is shown below.



Saving Changes and Exit

When completed the system configuration changes, select this option to leave Setup and reboot the computer so the new system configuration parameters can take effect. Select Exit Saving Changes from the Exit menu and press <Enter>.

Discarding Changes and Exit

Select this option to quit Setup without making any permanent changes to the system configuration. Select Exit Discarding Changes from the Exit menu and press <Enter>.

Discard Changes

Select Discard Changes from the Exit menu and press <Enter>.

Load Optimal Defaults

Automatically sets all Setup options to a complete set of default settings when select this option. Select Load Optimal Defaults from the Exit menu and press <Enter>.

Load Fail-Safe Defaults

Automatically sets all Setup options to a complete set of default settings when yselect this option. The Fail-Safe settings are designed for maximum system stability, but not maximum performance. Select the Fail-Safe Setup options if computer is experiencing system configuration problems.

Select Load Fail-Safe Defaults from the Exit menu and press <Enter>.

Chapter 4 Programming Guide

4.1 Reset to Default Information

```
//
// Portwell Confidential !
// Portwell Intellectual Property, All rights reserved.
//
/////////////////////////////////////////////////////////////////
//
// Program : 3727RSTD.CPP
// Descript. : PPAP-3727 Reset to Default test program
// Designer : Frank Hsu
// Language : Borland C++ 5.02
// O.S. : MS-DOS/Win98 only
// Update : 11222006 Release
//
//
/////////////////////////////////////////////////////////////////
/*
=====
//

; Reset to default status can be read from ESB2(631x)_ICH_GPI24.
; After Power On reset, GPI24 = low ( 0 )
; If Reset to Default (RST2DF) Button pressed ( Triggered )
; ,then GPI24 will be latch to high ( 1 ).
;
; RST2DF register can be cleared by 6300ESB_ICH_GPO18.
; Write a pulse timing ( High1_low_high2 ) to clear RST2DF to 0.
; High1 : output GPO18 high , and retain 15 us.
; Low : output GPO18 low , and retain 15 us.
; High2 : output GPO18 high again , and retain high always.
;
;
; Programming Guide :
;
;
; PG_Step1 : Enable GPIO IO function and get GPIOBASE
;
;
; How to program GPIO18 ( Output only , i.e. GPO18 )
; -----
; Get GPIOBASE =: B0:D31:F0:Offset[48..4Bh] ;(and let bit0 = 0 )
; GPIO_CNTL =: B0:D31:F0:Offset_4Ch_bit4P1 ;Enable 631xESB_ICH GPIO
;
; GPIO18
; GP_LVL (=:(GPIOBASE + 0Ch))_bit18P[0/1]; Write value 0/1
; -----
;
; How to read GPI24
; =====
; GPI24 status MUST NOT be inverted First.
; GPI_INV (=GPIOBASE+2Ch)-bit24P0. ( GPI24 not inverted )
;
; Get GPI24 status from GP_LVL (=GPIOBASE + 0Ch)-bit24
; 0 = low , 1= high level
; =====
*/
=====
```



```

#include "stdlib.h"
#include "conio.h"
#include "stdio.h"
#include "dos.h"           // for delay(), and sleep()

#pragma inline             // for inline asm , Need TASM.exe

#define GP_LVL_OFFSET 0x0C // The offset value from GPIOBASE
#define GPIO_USE_SEL_OFFSET 0x00 // offset from GPIOBASE ( Bit19P1 to define GPIO19)
#define GPO_BLINK_OFFSET 0x18 // Offset from GPIOBASE ( Bit19P0 , no blink )

#define GPO18_BYTE_CONVERT 0x02 // Convert Bit18 to ( GP_LVL+OFFSET + 02 ) for byte access
#define GPI24_BYTE_CONVERT 0x03 // Convert Bit18 to ( GP_LVL+OFFSET + 02 ) for byte access

#define PCR_ADDR 0x0CF8 // IO port CF8h for PCI config R/W
#define PCR_DATA 0x0CFC // IO port CFCh for PCI config R/W

#define OR_BIT2 0x04 // Bit2P1
#define AND0_BIT2 0xFB // Bit2P0

#define GPIOBASE_PCR_OFFSET 0x8000F848 // B0:D31:F0:Offset_48h
#define GPIOBASE_CNTL_PCR_OFFSET 0x8000F84C // B0:D31:F0:Offset_4Ch , bit4P1 to enable GPIO
#define IGNORE_BIT2T0 0xFFFFFFFF // Ignore Bit[2:0] for IO BAR
#define MASK_BIT4_P1 0x00000010 // Mask bit4 and write 1

#define portb 0x61
#define refresh_status 0x10

// Global Variable

unsigned int GPIOBASE ;

void IO_delay()
{
    inportb(0x80) ;
    inportb(0x80) ;
}
unsigned long inportd(unsigned short p)
{
    asm mov dx, p;
    asm in eax,dx ;
    return _EAX;
} // end of inportd

void outportd(unsigned short p, unsigned long v)
{
    asm mov dx, p;
    asm mov eax,v;
    asm out dx,eax;
} // end of outportd

void fixdelay_15us ()
{
    // delay 15 us
    unsigned char char_ah,char_al ;
    char_ah = inportb ( portb ) & refresh_status ;
fixdelay_loop :
    char_al = inportb ( portb ) & refresh_status ;
    if(char_ah == char_al ) goto fixdelay_loop ;
} // end of fixdelay_15us

void display_menu()
{

```



```

printf("\nPORTWELL PPAP-3727,3727RSTD.exe, V1.00 11-22-2006,All rights reserved.\n\n");
printf("    For PPAP-3727 Reset-to-Default test \n\n");
printf("    This status bit = 0 ---> Normal.    \n");
printf("    This status bit = 1 ---> RST2DF button has been pressed.\n");
printf("    This status bit can be read by 631xESB_ICH_GPI24, \n");
printf("    and can be cleared by an ICH_GPO18 High1-Low-High2 pulse.\n");
printf("    High1 = 30us High level    \n");
printf("    Low  = 30us Low level    \n");
printf("    High2 = High level again and no level change from now on.\n");
} // end of display_menu()

void GPO18_HLH()
{
    unsigned char al_char ;

    al_char = inportb( GPIOBASE + GP_LVL_OFFSET + GPO18_BYTE_CONVERT ) | OR_BIT2 ; // Bit2 P1
    outportb ( GPIOBASE + GP_LVL_OFFSET + GPO18_BYTE_CONVERT , al_char ) ; // output GPO18 1

    fixdelay_15us () ;
    fixdelay_15us () ;

    al_char = al_char & AND0_BIT2 ;
    outportb ( GPIOBASE + GP_LVL_OFFSET + GPO18_BYTE_CONVERT , al_char ) ; // output GPO18 0

    fixdelay_15us () ;
    fixdelay_15us () ;

    al_char = al_char | OR_BIT2 ; // Bit2 P1
    outportb ( GPIOBASE + GP_LVL_OFFSET + GPO18_BYTE_CONVERT , al_char ) ; // output GPO18 1

    fixdelay_15us() ;
    fixdelay_15us() ;

} // end of GPO18_HLH()

unsigned int main ()
{
    unsigned char al_char , al_ch , al_error =0x00 ;
    unsigned int j;
    unsigned long int i_EAX ;

    display_menu() ;

    // ----

    outportd(PCR_ADDR, GPIOBASE_CNTL_PCR_OFFSET ) ; // Enable GPIO
    i_EAX = inportd(PCR_DATA) | MASK_BIT4_P1 ; // Bit4P1 to enable GPIO
    outportd(PCR_ADDR, GPIOBASE_CNTL_PCR_OFFSET ) ;
    outportd(PCR_DATA, i_EAX ) ;

    outportd(PCR_ADDR, GPIOBASE_PCR_OFFSET ) ; // Get GPIOBASE
    GPIOBASE = inportd(PCR_DATA) & IGNORE_BIT2T0 ; // Truncate high word

    // ---

    // Define GPIO function for GPIO18 pin.
    al_char = inportb( GPIOBASE + GPIO_USE_SEL_OFFSET + GPO18_BYTE_CONVERT ) | OR_BIT2 ;
    outportb ( GPIOBASE + GPIO_USE_SEL_OFFSET + GPO18_BYTE_CONVERT , al_char ) ;

    // ===== MUST DO =====Start
    // Define GPO18 as non blinking

```

```

al_char = inportb( GPIOBASE + GPO_BLINK_OFFSET + GPO18_BYTE_CONVERT ) & AND0_BIT2 ;
outportb ( GPIOBASE + GPO_BLINK_OFFSET + GPO18_BYTE_CONVERT , al_char ) ;
// ===== MUST DO =====End

/*
; Testing way :
; --- t1
; Read GPI24 first , GPI24=0 ? if yes,pass ; if no, failed
;
; --- t2
; RST2DF button pressed and released , read GPI24 ,GPI24 = 1 ? if yes, pass ; if no, failed
;
; --- t3
; Clear RST2DF status to 0 ,read GPI24 ,GPI24 = 0 ? if yes, pass ; if no, failed
;----- t_start
*/

al_char = inportb( GPIOBASE + GP_LVL_OFFSET + GPO18_BYTE_CONVERT ) | OR_BIT2 ; // Bit2 P1
outportb ( GPIOBASE + GP_LVL_OFFSET + GPO18_BYTE_CONVERT , al_char ) ; // output GPO18 1 first

fixdelay_15us () ;

al_char = inportb( GPIOBASE + GP_LVL_OFFSET + GPI24_BYTE_CONVERT ) & 0x01 ;

if(al_char != 0x00)
{
    // Wrong initial status
    printf("\n Error#01:\nReset-to-Default F/F Initialization\n" Failed. ***** \n") ;
    printf(" \ ( This may be a H/W error or Reset-to-Default button has ever been pressed ! \) \n");
    al_error = al_error | 0x01 ;
    printf("\n\n Press \"Q\" key to stop test and return to DOS; or other key to go on next test") ;
    j = getche() ;
    al_ch = j ; // Truncate high byte
    if ( ( al_ch == 'Q' ) || (al_ch == 'q' ) ) goto test1_failed ;
}

// end_test1 :

printf("\n Press the Reset-to-Default button and then release it for the test NOW!\n");
printf("\n Ready ? If yes , then Press any key to start test ..... ") ;
getche() ;

al_char = inportb( GPIOBASE + GP_LVL_OFFSET + GPI24_BYTE_CONVERT ) & 0x01 ;
if( al_char != 0x01 )
{
    // Wrong reset to default
    printf("\n Error#02:\nReset-to-Default event latched by F/F \n" Failed. ***** \n") ;
    al_error = al_error | 0x02 ;
    printf("\n\n Press \"Q\" key to stop test and return to DOS; or other key to go on next test") ;
    j = getche() ;
    al_ch = j ; // Truncate high byte
    if ( ( al_ch == 'Q' ) || (al_ch == 'q' ) ) goto test_failed ;
}

// end_test2

GPO18_HLH() ;

al_char = inportb( GPIOBASE + GP_LVL_OFFSET + GPI24_BYTE_CONVERT ) & 0x01 ;
if ( al_char != 0x00 )
{
    // Wrong status after GPO18 H_L_H command
    printf("\n Error#03:\nClear Reset-to-Default F/F status \n" Failed. ***** \n") ;
    al_error = al_error | 0x04 ;
}

```

```

}

if(al_error != 0x00 ) goto test_failed ;
printf("\n\n <<..... PPAP-3727 RESET-TO-DEFAULT test OK ..^_...>> \n\n") ;
exit(0) ;

test1_failed :
GPO18_HLH() ; // Init F/F to 0 ( Normal state )
printf("\n\n <<***** PPAP-3727 RESET-TO-DEFAULT test FAIL *****>> ,error=0x%2X\n",al_error) ;
exit(1) ;

test_failed :
printf("\n\n <<***** PPAP-3727 RESET-TO-DEFAULT test FAIL *****>> ,error=0x%2X\n",al_error) ;
exit(1) ;

} // end of main

```

4.2 Bypass WDT Programming Guide

ABN-478 bypass and BYPASS Programming Guide.

0. Release note:	44
1. Bypass State Diagram	
2. System flow description:	45
2-1. System status diagram	45
2-2. Mode change diagram	46
2-3. Non-normal mode diagram	47
2.4 Next boot setting Diagram	48
2.5 Watch dog timer Diagram	49
3. Environment:	49
4. Check software component	49
5. Setup sequence:	50
5-1. Make execute file to test	50
6. Test program description	50
7. API description	51
7-1 dev_found	51
7-2. set_to_normal	51
7-3. set_to_nonnormal	51
7-4. set_nextboot	52
7-5. set_bpe	52
7-6 set_period_wdt	53
7-7. arm_wd	53
7-8. dis_arm_wdt_to_sts	54
7-9. dis_bp_wdt	55
7-10 read_status_now	55
7-11 read_settint_wdt	56
7-12 read_mode_nb	56
7-13 read_mode_pfwde	57
7-14 scenario_go	57
7-15 bypass_proc_step	58

4.2.0 Release note:

Version	Date	Description	Author
1.00	2007/3/9	Release origin version.	Angus Yang

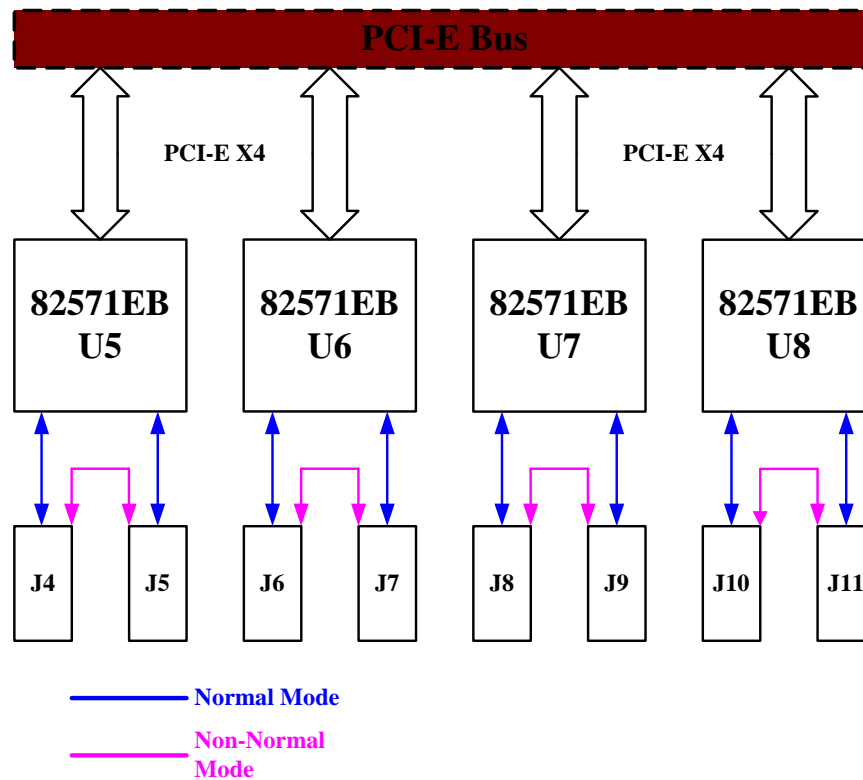


Figure 0 ABN-478 Block Diagram

4.2.1 Bypass State Diagram

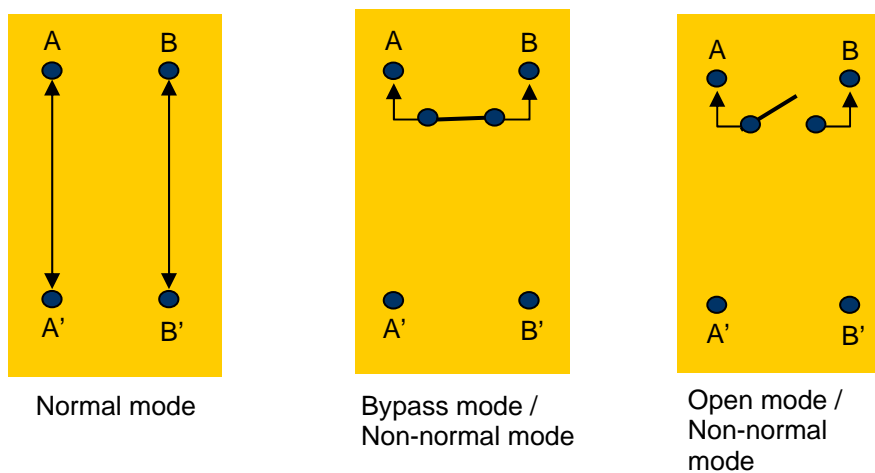


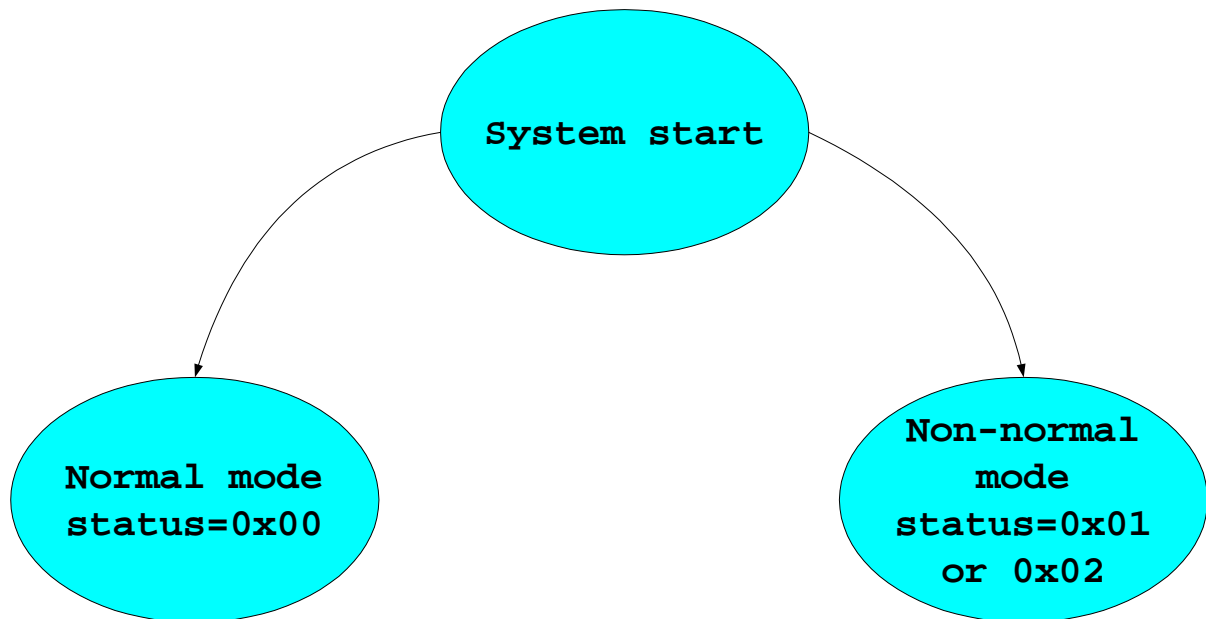
Figure 1

4.2.2 System flow description:

4.2.2-1. System status diagram

When power start up, system will go into normal mode or non-normal mode as Figure 2-1.

System start diagram



`status=0x00` (normal mode)

`status=0x01` (open mode)

`status=0x02` (bypass mode)

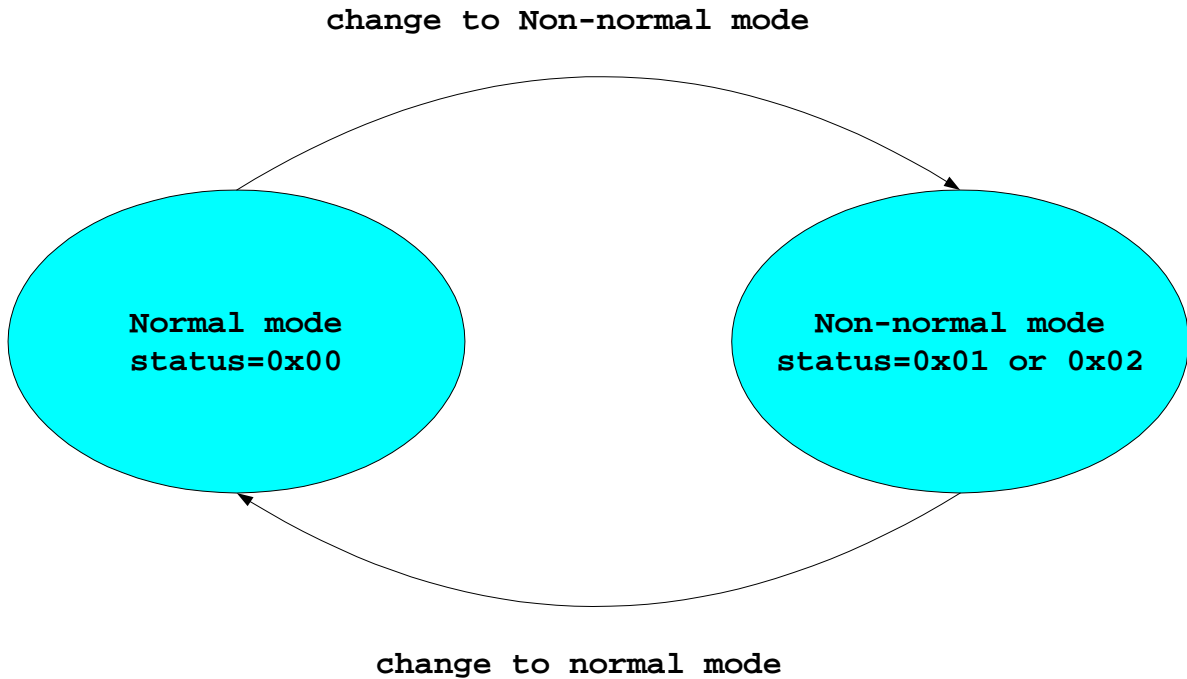
Figure 2-1.

When system starts, it will go into normal mode or non-normal mode. Which mode system will go into, it depends on the set by API **set_nextboot** (please refer to chapter 4.2.7-4)

4.2.2-2. Mode change diagram

Mode can be changed by software, its status flow as figure 2-2 descript

Mode change Diagram



`status= 0x00 (normal mode)`
`status= 0x01 (open mode)`
`status= 0x02 (bypass mode)`

Figure 2-2

Mode is changed to Non-normal mode by API **set_to_non-normal** (refer to chapter 4.2.7-3),
And it can be changed to normal mode by API **set_to_normal** (refer to chapter 4.2.7-2).

About status descript as below:

0x00, its mean is normal mode,

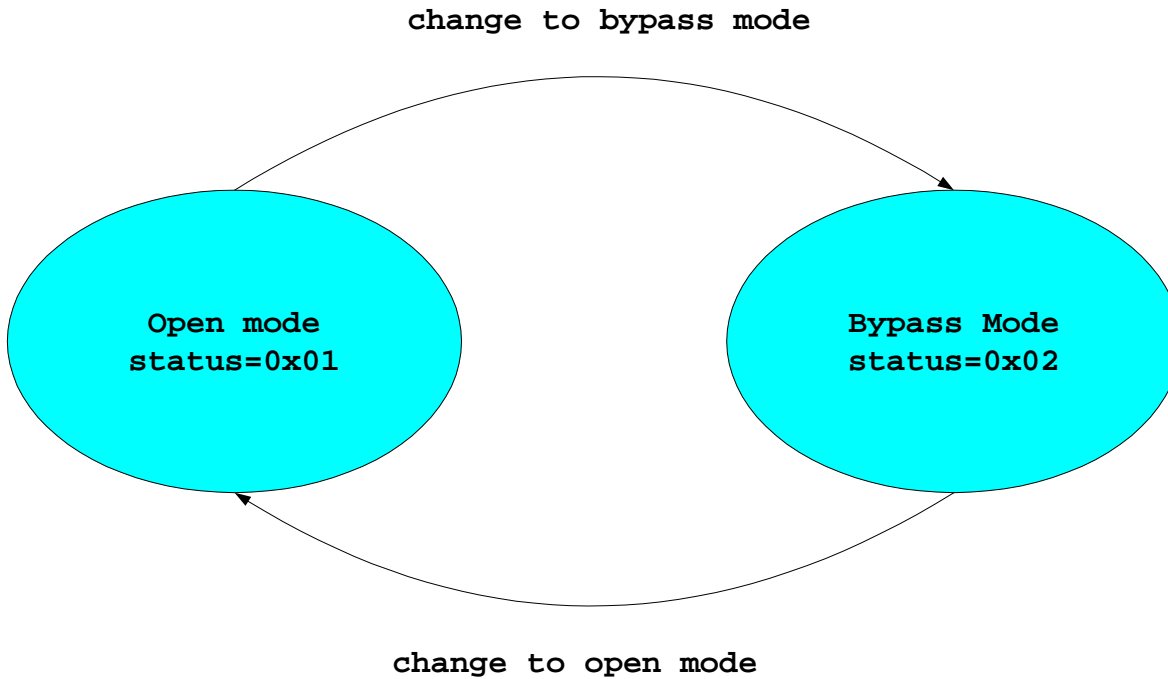
0x01, its mean is open mode.

0x02, its mean is bypass mode.

4.2.2-3. Non-normal mode diagram

About non-normal mode, it includes two parts, one is open mode, and another is bypass mode. The detail description as below

Non-normal mode Diagram



`status= 0x01 (open mode)`
`status= 0x02 (bypass mode)`

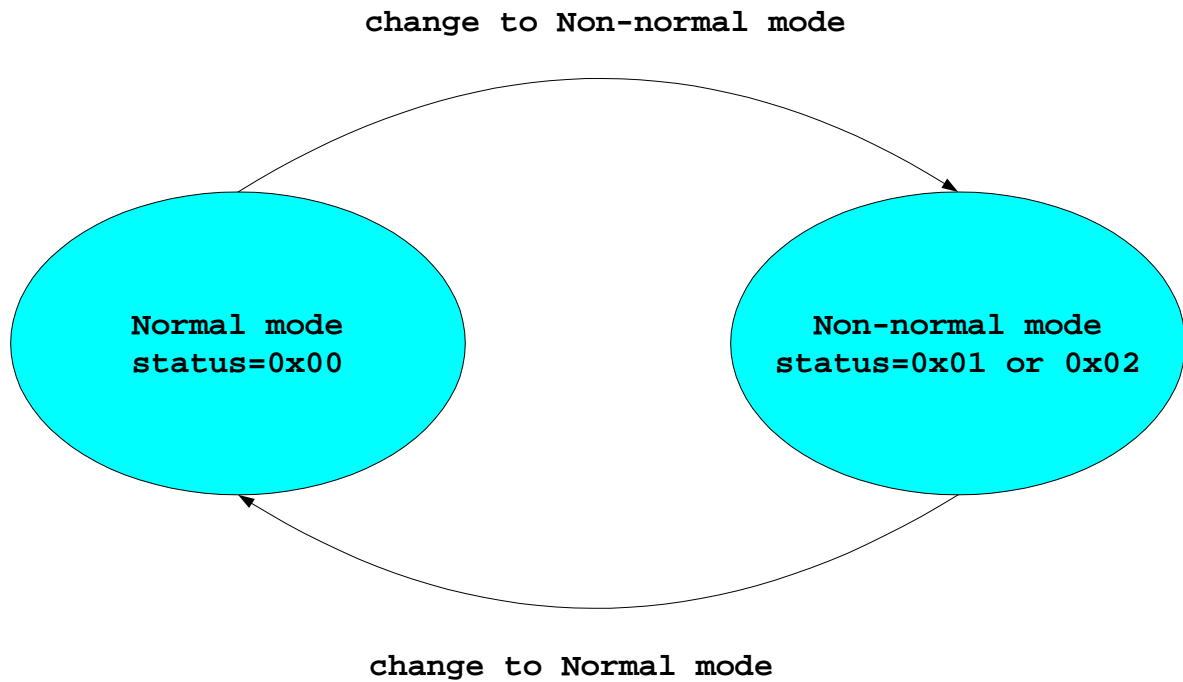
Figure-2.3

In non-normal mode. Mode can be changed to open mode or bypass mode. They can be set by API `set _bpe` (refer to chapter 4.2.7-5). If user wants to assign system to open mode, send command `set_bpe` as `SET_BPE_OPEN`, assign system to bypass mode, send command `set_bpe` as `SET_BPE_BYPASS`.

4.2.2-4 Next boot setting Diagram

About next boot setting, it descript as below (to see figure 2-4)

Next boot setting diagram



`status=0x00` (normal mode)
`status=0x01` (open mode)
`status=0x02` (bypass mode)

Figure 2-4

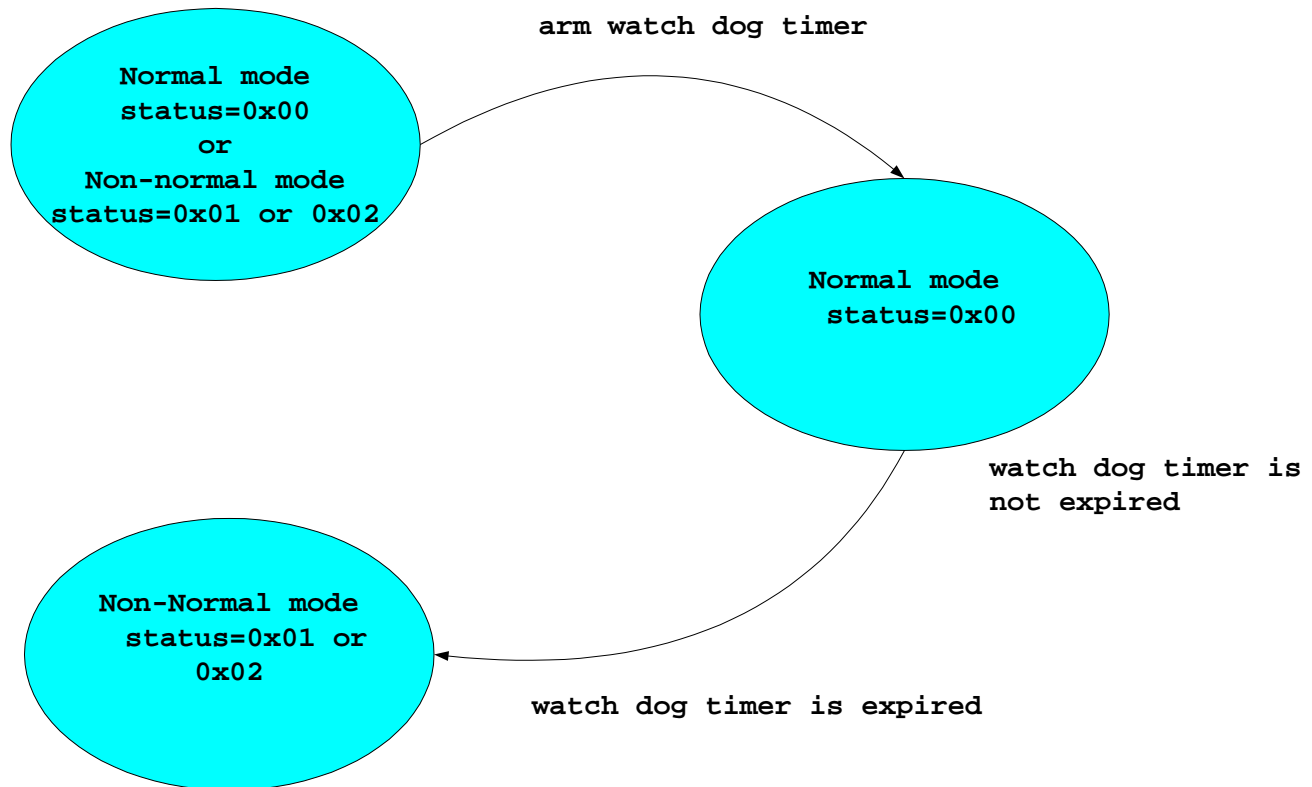
User can assign system be in normal mode or non-normal when system start up.
Set it by API **set_nextboot** (refer to chapter 4.2.7-4).

If user wants to assign system to normal mode, set **set_nextboot** as **SET_NB_NORMAL**.
If user wants to assign system to non-normal mode, set **set_nextboot** as **SET_NB_NON_NORMAL**.

4.2.2-5 Watch dog timer Diagram

About Watch dog timer status, the detail descript as below

ARM_Watch dog timer Diagram



status= 0x00 (normal mode)
status= 0x01 (open mode)
status= 0x02 (bypass mode)

Figure 2-5

When watch dog timer is armed, system will be in normal mode, and waiting for watch dog timer expiring, after watch dog timer expire, system will be in non-normal mode.

User can set it by API **arm_wdt** (refer to chapter 4.2.7-7)

4.2.3. Environment:

We build and test programs base on gcc version 4.1.0 20060304 (Fedora Core 5).
And gcc version 3.2.2 20030222(red-hat 9)

4.2.4. Check software component

- ./lib/portwell_bypass.a
- Test program (. /ap/)
It includes such test program as below://v1.01


```

test_all.c
set_normal.c
set_nonnormal.c
test_long_api.c
test_nb_nonnormal_mode.c
test_nb_normal_mode.c
test_pfwde_bypass.c
test_pfwde_open.c
test_period_wdt_1.c
test_period_wdt_2.c
test_period_wdt_1691252.c
test_period_wdt_2114065.c

```

*note: all functions are all for single card test,it does not support more cards test at the same time now.

4.2.5. Setup sequence:

4.2.5-1. Make execute file to test

Please go into abn478 path, and then do ./compile.sh command to build execute file to test.

4.2.6. Test program description

item	Function name	Description
1	test_all.c	Test all functions as above description
2	Set_normal.c	Change mode to normal
3	Set_nonnormal.c	Change mode to open
4	test_pfwde_open.c	Set to open mode when power off
5	test_pfwde_bypass.c	Set to bypass mode when power off
6	test_nb_normal_mode.c	Set system to normal mode when next boot.
7	test_nb_nonnormal_mode.c	Set system to non-normal mode when next boot.
8	test_period_wdt_1.c	set watch dog timer period as 1 sec
9	test_period_wdt_2.c	set watch dog timer period as 2 sec
10	test_period_wdt_3.c	set watch dog timer period as 3 sec
11	test_period_wdt_4c	set watch dog timer period as 4 sec

4.2.7-1 dev found

Function name	dev_found
Description	To find the NIC device
Format:	
int dev_found(unsigned char prod_num)	
Input: prod_num	
1-1. PROD_ABN478: bypass card of abn478	
Return:	
0: device was found successfully	
-1: can not find device.	

4.2.7-2. set to normal

Function name	set_to_normal
Description	Set Ethernet segment to normal mode
Format:	
unsigned char set_to_normal(unsigned char mode,unsigned char seg,unsigned char proc_dev)	
Input:	
1. mode:0x00 ignore.	
2. seg:	
1:segment 1.	
2:segment 2.	
3:segment 3.	
4:segment 4.	
3. proc_dev:	
3-1. PROD_ABN478: bypass card of abn478	
Return: 0: return ok.	
-1: return fail.	

4.2.7-3. set to nonormal

Function name	set_to_nonormal
Description	Set Ethernet segment to non-normal mode
Format:	
unsigned char set_to_nonormal(unsigned char mode,unsigned char seg,unsigned char proc_dev)	
Input:	
1. mode:0x00 ignore.	
2. seg:	
1:segment 1.	
2:segment 2.	
3:segment 3.	
4:segment 4.	
3. proc_dev:	
3-1. PROD_ABN478: bypass card of abn478	
Return: 0: return ok.	

-1: return fail.

4.2.7-4. set_nextboot

Function name	set_nextboot
Description	1. Set Ethernet segment to normal mode or non-normal mode when boot up next time
Format:	
unsigned char set_nextboot(unsigned char mode,unsigned char seg,unsigned char proc_dev)	
Input:	
1. mode:	
SET_NB_NORMAL : normal when next boot up.	
SET_NB_NON_NORMAL:non_normal when next boot up	
2. seg:	
1:segment 1.	
2:segment 2.	
3:segment 3.	
4:segment 4.	
3. proc_dev:	
3-1. PROD_ABN478: bypass card of abn478	
Return: 0: return ok.	
-1: return fail.	

4.2.7-5. set_bpe

Function name	set_bpe
Description	Set Ethernet segment to open mode or bypass mode when power fails or

	watch dog timer expires
Format:	
unsigned char set_bpe(unsigned char mode,unsigned char seg,unsigned char proc_dev)	
Input:	
1. mode:	
SET_BPE_OPEN: set Ethernet port to open mode.	
SET_BPE_BYPASS: set Ethernet port to bypass mode.	
2. seg:	
1:segment 1.	
2:segment 2.	
3:segment 3.	
4:segment 4.	
3. proc_dev:	
3-1. PROD_ABN478: bypass card of abn478	
Return: 0: return ok.	
-1:return fail.	

4.2.7-6 set_period_wdt

Function name	set_period_wdt
Description	Set watch dog timer period
Format:	
int set_period_wdt (unsigned char pd,unsigned char seg,unsigned char proc_dev)//v1.04	
Input:	
1. pd	
1:set watch dog timer period as 1s	
2:set watch dog timer period as 2s	
3:set watch dog timer period as 3s	
4:set watch dog timer period as 4s	
(Abn478 supports watch dog timer count from 1 to 63 seconds)	
1. seg:	
1:segment 1.	
2:segment 2.	
3:segment 3.	
4:segment 4.	
2. proc_dev:	
3-1. PROD_ABN478: bypass card of abn478	
Return: 0: return ok.	
-1:return fail.	

4.2.7-7. arm wd

Function name	arm_wdt
Description	Test mode changing rate by watchdog timer, and its period is set by set_period_wdt function (6-6). Its mode will be changed to bypass-mode when watchdog timer time out.
Format:	
unsigned char arm_wdt(unsigned char mode,unsigned char seg,unsigned char proc_dev) Input: 1. mode: 0x00 ignore 2. seg: 1:segment 1. 2:segment 2. 3:segment 3. 4:segment 4. 3. proc_dev: 3-1. PROD_ABN478: bypass card of abn478 Return: 0: return ok. -1: return fail.	

4.2.7-8. dis arm wdt to sts

Function name	Dis_arm_wdt_to_sts
Description	Clear the set of watchdog timer.
Format:	
unsigned char dis_arm_wdt_to_sts(unsigned char mode,unsigned char seg,unsigned char proc_dev) Input: 1. mode: 0x00 ignore 2. seg: 1:segment 1. 2:segment 2. 3:segment 3. 4:segment 4. 3. proc_dev: 3-1. PROD_ABN478: bypass card of abn478 Return: 0: return ok. -1: return fail.	

4.2.7-9. dis bp wdt

Function name	dis_bp_wdt
Description	Disable watch dog timer and keep status as current state
Format:	
1. If Ethernet is on bypass mode, when user does this command, it will be disable watch dog and keep it in bypass mode. If normal mode, after doing this command, it will be kept in normal mode.	
unsigned char dis_bp_wdt(unsigned char mode,unsigned char seg,unsigned char proc_dev)	
Input:	
1. mode:	
0x00 ignore	
2. seg:	
1:segement 1.	
2:segement 2.	
3:segement 3.	
4:segement 4.	
3. proc_dev:	
3-1. PROD_ABN478: bypass card of abn478	
Return: 0: return ok.	
-1:return fail.	

4.2.7-10 read status now

Function name	read_status_now
Description	Read bypass status
Format:	
unsigned char read_status_now(unsigned char mode,unsigned char seg,unsigned char proc_dev)	
Input:	
1. mode:	
0x00 ignore	
2. seg:	
1:segement 1.	
2:segement 2.	
3:segement 3.	
4:segement 4.	
3. proc_dev:	
3-1. PROD_ABN478: bypass card of abn478	
Return: 0: normal mode.	
1: open mode	
2:bypass mode	

4.2.7-11 read settint wdt

Function name	read_setting_wdt
Description	Read watch dog timer setting
Format:	
unsigned char read_setting_wdt(unsigned char mode,unsigned char seg,unsigned char proc_dev)	
Input:	
1. mode:	
0x00 ignore	
2. seg:	
1:segement 1.	
2:segement 2.	
3:segement 3.	
4:segement 4.	
3. proc_dev:	
3-1. PROD_ABN478: bypass card of abn478	
Return:	
0: ok.	
1: fail.	
0xff:device fail.	

4.2.7-12 read mode nb

Function name	read_mode_nb
Description	Read next boot status
Format:	
unsigned char read_mode_nb(unsigned char mode,unsigned char seg,unsigned char proc_dev)	
Input:	
1. mode:	
0x00 ignore	
2. seg:	
1:segement 1.	
2:segement 2.	
3:segement 3.	
4:segement 4.	
3. proc_dev:	
3-1. PROD_ABN478: bypass card of abn478	
Return:	
0: next boot=normal mode.	
1: next boot=non-normal mode & BPE=open mode.	
2: next boot=non-normal mode & BPE=bypass mode.	

4.2.7-13 read_mode_pfwdte

Function name	read_mode_pfwdte
Description	Read bpe status
Format:	
unsigned char read_mode_pfwdte(unsigned char mode,unsigned char seg,unsigned char proc_dev)	
Input:	
1. mode:	
0x00 ignore	
2. seg:	
1:segement 1.	
2:segement 2.	
3:segement 3.	
4:segement 4.	
3. proc_dev:	
3-1. PROD_ABN478: bypass card of abn478	
Return:	
1: pfwdte to open mode.	
2: pfwdte to bypass mode.	

4.2.7-14 scenario_go

Function name	scenario_go
Description	Go abn478 bypass module all functions automatically.
Format:	
unsigned char	
scenario_go(unsigned int rst[[DO_FUNCTION_NUM],unsigned char prod_type)	
Input:	
1. rst:	
array of return code.	
0:normal mode.	
1:open mode.	
2:bypass mode.	
2. prod_type:	
2-1. PROD_ABN478: bypass card of abn478	
DO_FUNCTION_NUM:0x0c.(it has 12 functions to test)	
Return:	
0:ok	
1:fail.	

4.2.7-15 bypass_proc_step

Function name	bypass_proc_step
Description	Run abn478 bypass function
Format:	
unsigned char bypass_proc_step(unsigned char dowhat,unsigned char *rst,unsigned char prod_type,unsigned char para_var)Input:	
1. dowhat:	
DO_SET_NORMAL: do normal mode function	
DO_SET_NONNORMAL:do non_normal mode function	
DO_SET_NEXTBOOT_NONNORMAL:do next boot non_normal mode function.	
DO_SET_NEXTBOOT_NORMAL:do next boot normal mode function	
DO_SET_BPE_OPEN:record bpe mode to open	
DO_SET_BPE_OPEN_RUNTIME:change bpe mode to open mode(relay will action)	
DO_SET_BPE_BYPASS:record bpe mode to bypass mode.	
DO_SET_BPE_BYPASS_RUNTIME:change bpe mode to bypass mode(relay will action)	
DO_SET_WDT_PERIOD:set watch dog timer period.	
DO_SET_WDT_DIS_BP: disable watch dog timer.	
DO_SET_WDT_DIS_ARM: Clear the set of watchdog timer.	
DO_SET_WDT_ARM:do watch dog timer expire function.	
2. *rst	
array of return code.	
2-1. 0:normal mode.	
2-2. 1:open mode.	
2-3. 2:bypass mode.	
3. prod_type:	
3-1. PROD_ABN478: bypass card of abn478	
DO_FUNCTION_NUM:0x0c.(it has 12 functions to test)	
Return:	
0:ok	
1:fail.	

4.3 About EZIO2

Proprietary keypad and LCD display interfaces are implemented in traditional computing system design, but they are usually different from system to system. The main purpose to roll this module out is to provide an easier human-machine interface for those computing systems regarding application friendly operation as a “must.”

The design goals of this interface are:

- ◆ A single interface for those applications where both LCD display and keypad are required.
- ◆ This interface should be available in every computing system.
- ◆ The communication implementation should be OS independent.

Our solution is to use “Serial port” as the interface for both LCD display and keypad. A simple protocol is further defined so that applications can directly communicate with this module no matter what the Operating System is.

WARNING!

THE LCD DRIVER ICS ARE MADE OF CMOS PROCESS, DAMAGED BY STATIC CHARGE VERY EASILY. MAKE SURE THE USER IS GROUNDED WHEN HANDLING THE LCD.

4.3.1 Features

- Ideal user interface for communication appliance
- No driver required; OS independent
- Alphanumeric characters display support
- Four key pads can be customized for different applications
- Easy system installation and operation
- Clearly display system status
- Single interface to SBC or M/B

4.3.2 Technical Support Information

For further support, users may also contact Portwell’s headquarter in Taipei or your local distributors.

Taipei Office Phone Number: +886-2-27992020

4.3.3 Mechanical Specification

Module Size (mm):	<ul style="list-style-type: none"> 101.6(W) x 26.0(H) x 30.6(D) (max.)
Display Format:	<ul style="list-style-type: none"> 16 characters x 2 lines
Character Size:	<ul style="list-style-type: none"> 3.0 x 5.23 mm

4.3.4 General Specification

General Specification

Display Resolution:	<ul style="list-style-type: none"> 16 characters x 2 lines
Dimensional Outline (mm):	<ul style="list-style-type: none"> 101.6(W) x 26.0(H) x 30.6(D) (max.)
Function Key:	<ul style="list-style-type: none"> Four operation keys (up, down, enter and ESC)
Display Icon:	<ul style="list-style-type: none"> Eight self-defined icons
Interface:	<ul style="list-style-type: none"> RS-232

Absolute Maximum Rating

Item	Normal Temperature			
	Operating		Storage	
	Max.	Min.	Max.	Min.
Ambient Temperature	0°C	+45°C	-20°C	+70°C
Humidity (w/o condensation)	Note 2, 4		Note 3, 5	

4.3.5 Product Outlook



4.3.6 Interface Pin Assignment

There are only two connectors in this module, as shown in **Figure C.2-1**: power connector and Serial Port connector. The power source into this module is 5 volt only. There are only three pins used in the Serial Port interface (**Figure C.2-2**).

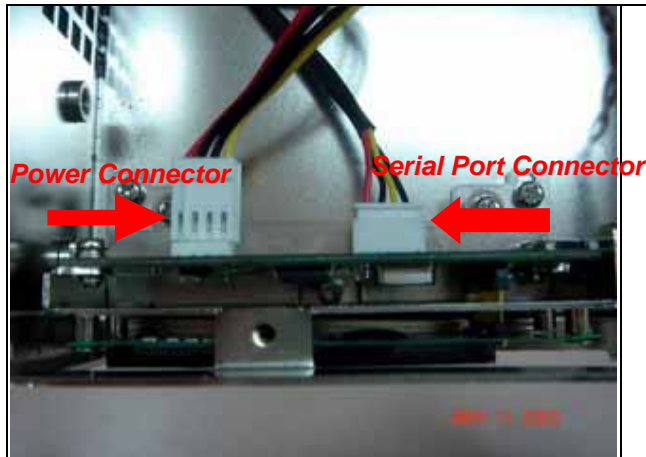


Fig. C.2-1 Power connector and serial port connector of EZIO-100

5	4	3	2	1
10	9	8	7	6

Pin 2: TxD Pin 3 : RxD Pin 5 : Ground

Fig. C.2-2 Pin assignment

In other words, the Serial Port is defined as DCE. Therefore, we can use a straight-through cable to connect it to the Serial Port of most of the computers, defined as DTE.

(1) Interface Pin Assignment

PIN NO.	PIN OUT	Description
1	NC	No connector
2	RXD	RS232 Data
3	TXD	RS232 Data
4	NC	No connector
5	V _{SS}	Ground
6	NC	No connector
7	NC	No connector
8	NC	No connector
9	NC	No connector
9	NC	No connector

(2) Power

PIN NO.	PIN OUT	Description
1	NC	No connector
2	GND	Power GND
3	GND	Power GND
4	+5V	Power VCC (+5V)

4.3.7 Hardware installation

The installation steps are:

- ◆ Connect the power connector to the power connector of this module.
Connect the straight-through cable between Serial Port of this module and computer

4.3.8 EZIO2 Function Command

First, all versions (00A, 01A, 02A) of EZIO can use those commands. Only the 02A version of EZIO firmware that adds “FE 28” & “FE 37” command can control start of HEX & End of HEX.

EZIO is an intelligent device, which will display those data received from RS-232 port and reply key pressing status to polling command from RS-232 port. Both commands and data go thru RS-232 ports. To distinguish between data and commands, the LCD/key-pad module recognizes a command prefix, 254 (Hex 0FE). The byte following “254” will be processed as a command. For example, to clear the screen, send the command prefix (254) followed by the LCD clear-screen code (1). The valid data range is shown as the following table:

<i>Valid data range</i>	<i>Displayed characters</i>
0-7	Customized icon 0-7
48-57 (30-39 Hex)	0-9
65-90 (41-5A Hex)	A-Z
97-122 (61-7A Hex)	a-z

To get the key pressing status, a “read key” command can be issued to this module, which will check the key-pressing status and reply accordingly. The following are the commands and corresponding Decimal/Hex values:

	<i>Functions/commands</i>	<i>Decimal/Hex</i>	<i>Comment</i>
1.	Start Of HEX	40/28	Only for 02A
2.	End Of HEX	55/37	Only for 02A
3.	Clear screen	1/01	
4.	Home cursor	2/02	
5.	Read key	6/06	See note 1
6.	Blank display (retaining data)	8/08	
7.	Hide cursor & display blanked characters	12/0C	
8.	Turn on (blinking block cursor)	13/0D	
9.	Show underline cursor	14/0E	
10.	Move cursor 1 character left	16/10	
11.	Move cursor 1 character right	20/14	
12.	Scroll 1 character left	24/18	
13.	Scroll 1 character right	28/1C	
14.	Set display address (position the cursor) location	128 (Hex080)+ Location	See note 2
15.	Set character-generator address	64 (Hex 040)+ address	See note 3

Note 1: Upon receiving the “read key” command from host computer, the LCD/keypad module will check the status of every key and reply with status command accordingly. The replied message from LCD/key-pad module consists of a header and a status byte. The header byte is 253 (Hex0FD). The high nibble (with the most significant bit) of the status byte is always “4” and the low nibble (with the least significant bit) of the status byte is used to indicate key pressing status of the keypad module. This nibble will be “F” (of four 1s), if no key pressed while the “read key” received. “0” will be used to indicate key pressing status of corresponding key. There are four keys in this module – upper arrow, down arrow, enter (ENT), and escape (ESC). The relationship between the function key, corresponding status bit and status byte is shown as the table below.

Function key	Corresponding status bit	Status byte
Escape	The fourth bit of lower nibble (the least significant bit) (1110)	B7 (H)
Up arrow	The third bit of lower nibble (1101)	BE (H)
Enter	The second bit of lower nibble (1011)	BB (H)
Down arrow	The first bit of lower nibble (0111)	BD (H)

More than one key can be pressed at the same time so that there may be more than one “0”s in the low nibble of status byte. For example, if Up and Down arrow keys are pressed at the same time while “read key” command received, the replied status will be “Hex045”.

Note 2: This command can be used to place the cursor at any location. The corresponding address for each character on the screen is as follows:

For 16x2 Display Address

Character	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Location	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
(Address)	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

The addresses of characters at the same row are continuous, so moving cursor commands can be applied to shift the cursor position back and forth. However, the addresses of characters between upper and lower row are discontinuous. To change cursor position between upper row and lower row, this command will be applied.

Note 3: This command can be used to create customized icon. The starting address is 64 and every character will take 8 bytes to create a 5(W) x 7(H) resolution picture, as shown below:

CG RAM MAPPING

CG RAM Address						Character Patterns (CG RAM data)										
5	4	3	2	1	0	7	6	5	4	3	2	1	0			
High			Low			High			Low							
0	0	0	0	0	0	*	*	*	0	1	1	0	0	Character Pattern		
			0	0	1				1	0	0	1	0			
			0	1	0				0	0	1	0	0			
			0	1	1				0	1	0	0	0			
			1	0	0				1	1	1	1	0			
			1	0	1				0	0	0	0	0			
			1	1	0				0	0	0	0	0			
			1	1	1				0	0	0	0	0			
0	0	1	0	0	0	*	*	*	1	1	1	1	1	Character Pattern		
			0	0	1				1	0	0	0	1			
			0	1	0				1	0	1	0	1			
			0	1	1				1	0	1	1	1			
			1	0	0				1	0	1	0	1			
			1	0	1				1	0	0	0	1			
			1	1	0				1	1	1	1	1			
			1	1	1				0	0	0	0	0			
									Cursor							

4.3.9 Character Generator ROM (CGROM)

Upper 4 bits Lower 4 bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	CG RAM (1)			0	0	P	\	P				—	マ	E	o	p
0001	CG RAM (2)		!	1	A	Q	a	a			u	ア	+	△	é	q
0010	CG RAM (3)		"	2	B	R	b	r			r	イ	ウ	×	p	e
0011	CG RAM (4)		#	3	O	S	c	c			J	ウ	テ	セ	æ	
0100	CG RAM (5)		*4	4	O	T	t	t			\	エ	ト	ホ	h	o
0101	CG RAM (6)		5	E	U	e	u				u	オ	★	1	é	Q
0110	CG RAM (7)		6	F	U	f	v				マ	カ	ニ	ヨ	p	z
0111	CG RAM (8)		7	G	W	g	w				ア	+	ズ	ラ	g	m
1000	CG RAM (1)		(8	H	X	h	x			ノ	ウ	ホ	リ	フ	ズ
1001	CG RAM (2))	9	I	V	i	v			ウ	ケ	ル	ル	フ	g
1010	CG RAM (3)		*#	J	Z	j	z				エ	コ	ン	ル	ジ	フ
1011	CG RAM (4)		+;	K	L	k	l				オ	サ	ヒ	ロ	フ	フ
1100	CG RAM (5)		, <	L	*	l	l				ホ	エ	フ	フ	é	m
1101	CG RAM (6)		— =	M	J	m)				エ	ズ	△	ウ	ト	÷
1110	CG RAM (7)		u	>	N	^	n	+			エ	セ	ホ	フ	フ	
1111	CG RAM (8)		/ ?	O	_	o	+				ウ	ウ	マ	フ	é	■

4.3.10 Sample Codes

```
/* *****
* EZIO RS232 LCD Control Sample Program
* *****
* *****
* Company:      Portwell Inc.
* Date:         4/16/2003
* Program:      02A.c
* Version:      1.02
* Compile:      Linux GNU C
* Purpose:      Direct access to EZIO LCD, the program will display
*               messages according to the control button. The current
*               version only has the following function:
*
*               1: display welcome message
*               2: display UP message if "scroll up" button is pressed
*               3: display ENTER message if "ENTER" button is pressed
*               4: display ESC message if "ESC" button is pressed
*               5: display DOWN message if "scroll down" button is pressed
*
* Program Overview:
*
*   - Parameters:
*       fd          : a file name for open() method, here represents the com port
*       Cmd         : command prefix
*       cls         : clear command
*       init        : initialize command
*       blank       : display blank screen
*       stopsend    : stop input/output
*       home        : move cursor to initial position
*       readkey     : set to read from EZIO
*       hide        : hide cursor & display blanked characters
*       movel       : move cursor one character left
*       mover       : move cursor one character right
*       turn        : turn on blinking-block cursor
*       show        : turn on underline cursor
*       scl         : scroll cursor one character left
*       scr         : scroll cursor one character right
*       setdis      : set character-generator address
*
*   - Procedure:
*       1. The program sets up the environment, i.e. com port settings.
*       2. The main function MUST call init() twice to initialize EZIO
*          before any communication.
*       3. For executing any command, the command prefix, Cmd, MUST be
*          called be command. So all command contains two parts, eg.
*          to initialize the sequence of HEX number is 0xFE, 0x25.
*       4. After clear screen and display welcome message, ReadKey()
*          method must be call to advise EZIO for reading data.
*       5. A pooling method is implemented to get input from EZIO while
*          any button is pressed.
*
*   - NOTE: This program is a sample program provided " AS IS" with NO
*           warranty.
*
* Copyright (c) Portwell, Inc. All Rights Reserved.
* *****/
```

```

#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>

static int fd;

void SetEnvironment () {
    system("stty ispeed 2400 < /dev/ttyS1");
    system("stty raw < /dev/ttyS1");
}

int Cmd = 254; /* EZIO Command */
int cls = 1; /* Clear screen */
void Cls () {
    write(fd,&Cmd,1);
    write(fd,&cls,1);
}

int init = 0x28;
void Init () {
    write(fd,&Cmd,1);
    write(fd,&init,1);
}

int stopsend = 0x37;
void StopSend () {
    write(fd,&Cmd,1);
    write(fd,&init,1);
}

int home = 2 ; /* Home cursor */
void Home () {
    write(fd,&Cmd,1);
    write(fd,&home,1);
}

int readkey = 6 ; /* Read key */
void ReadKey () {
    write(fd,&Cmd,1);
    write(fd,&readkey,1);
}

int blank = 8 ; /* Blank display */
void Blank () {
    write(fd,&Cmd,1);
    write(fd,&blank,1);
}

int hide = 12 ; /* Hide cursor & display blanked characters */
void Hide () {
    write(fd,&Cmd,1);
    write(fd,&hide,1);
}

int turn = 13 ; /* Turn On (blinking block cursor) */
void TurnOn () {
    write(fd,&Cmd,1);
    write(fd,&turn,1);
}

int show = 14; /* Show underline cursor */
void Show () {
    write(fd,&Cmd,1);
    write(fd,&show,1);
}

```

```

}

int movel = 16      ; /* Move cursor 1 character left */
void MoveL () {
    write(fd,&Cmd,1);
    write(fd,&movel,1);
}

int mover = 20      ; /* Move cursor 1 character right */
void MoveR () {
    write(fd,&Cmd,1);
    write(fd,&mover,1);
}

int scl = 24;      /* Scroll cursor 1 character left */
void ScrollL(){
    write(fd,&Cmd,1);
    write(fd,&scl,1);
}

int scr = 28;      /* Scroll cursor 1 character right */
void ScrollR(){
    write(fd,&Cmd,1);
    write(fd,&scr,1);
}

int setdis = 64; /* Command */
void SetDis(){
    write(fd,&Cmd,1);
    write(fd,&setdis,1);
}

}

/* Add or Change Show Message here */
char mes1[] = "Portwell EZIO";
char mes2[] = "*****";
char mes3[] = "Up is selected";
char mes4[] = "Down is selected";
char mes5[] = "Enter is selected";
char mes6[] = "ESC is selected";
char nul[] = "          ";

int a,b;
void ShowMessage (char *str1 , char *str2) {
    a = strlen(str1);
    b = 40 - a;
    write(fd,str1,a);
    write(fd,nul,b);
    write(fd,str2,strlen(str2));
}

int main () {

    SetEnvironment(); /* Set RAW mode */

    fd = open("/dev/ttyS1" ,O_RDWR);/** Open Serial port (COM2) */

    Init(); /* Initialize EZIO twice */
    Init();

    Cls(); /* Clear screen */
    ShowMessage(mes1,mes2);

```

```

while (1) {
    int res;
    char buf[255];

    SetDis();
    ReadKey(); /* sub-routine to send "read key" command */
    res = read(fd,buf,255); /* read response from EZIO */

    switch(buf[1]) {      /* Switch the Read command */

        case 0x4D : /* Up Botton was received */
            Cls();
            ShowMessage(mes1,mes3); /** display "Portwell EZIO" */
            break;          /** display "Up is selected" */

        case 0x47 :  /** Down Botton was received */
            Cls();
            ShowMessage(mes1,mes4); /** display "Portwell EZIO" */
            break;          /** display "Down is selected" */

        case 0x4B :  /** Enter Botton was received */
            Cls();
            ShowMessage(mes1,mes5); /** display "Portwell EZIO" */
            break;          /** display "Enter is selected" */

        case 0x4E :  /** Escape Botton was received */
            Cls();
            ShowMessage(mes1,mes6); /** display "Portwell EZIO" */
            break;          /** display "Escape is selected" */

    }
}
}

```

4.4 GPIO Sample Code

```
// Portwell Confidential !
// Portwell Intellectual Property, All rights reserved.
//
////////////////////////////////////
//
//   Program   : 3727GPIO.CPP
//   Descript. : PPAP-3727 GPIO test program
//   Designer  : Frank Hsu
//   Language  : Borland C++ 5.02
//   O.S.      : MS-DOS/Win98 only
//   Update    : 11222006 Release
//
//
////////////////////////////////////
//
//
// GPIO on PPAP-3727
//   PPAP-3727 J30_Pin1=GPIO1:from SUPER I/O_GPIO17
//       J30_Pin2=GPIO2:from SUPER I/O_GPIO16
//       J30_Pin3=GPIO3:from SUPER I/O_GPIO15
//       J30_Pin4=GPIO4:from SUPER I/O_GPIO14
//       J30_Pin6=GPIO5:from SUPER I/O_GPIO10
//       J30_Pin7=GPIO6:from SUPER I/O_GPIO11
//       J30_Pin8=GPIO7:from SUPER I/O_GPIO12
//       J30_Pin9=GPIO8:from SUPER I/O_GPIO13
//   <<<<< Be careful Pin5=GND , Pin10=VCC >>>>>
//
// Programming Guide :
// Step1 : CR29_Bit[7..6]P[0,1] to select GPIO10~17 pin
// Step2 : LD7_CR07h_P[07h] : point to LD7
// Step3 : LD7_CR30h_bit0_P1 : Enable LD7
// Step4 : Start 4 test items ( t1 , t2, t3, t4 )
//       LD7_CRF0h definition : GPIO17 ~ 10 : 1 = input , 0 = output pin
//       LD7_CRF2h_P[00h] : Always let CRF1 ( GPIO data port ) non-invert.
//       LD7_CRF1h : GPIO17~10 data port ,
//       t1 : GPO17,16,15,14 output [1,1,1,1] to GPI10,11,12,13
//       t2 : GPO17,16,15,14 output [0,0,0,0] to GPI10,11,12,13
//       t3 : GPO10,11,12,13 output [1,1,1,1] to GPI17,16,15,14
//       t4 : GPO10,11,12,13 output [0,0,0,0] to GPI17,16,15,14
//
//
#include "stdlib.h"
#include "conio.h"
#include "stdio.h"
#include "dos.h"           // for delay(), and sleep()

// Global constant ----- Start -----
#define config_W83627 0x2E // Hardware strapping
#define GPIO1_LDN 0x07 // W83627 GPIO1 LDN = 0x07 for GP10~17
#define CR07 0x07
```



```

#define CR29      0x29
#define CR30      0x30
#define CRF0      0xF0
#define CRF1      0xF1
#define CRF2      0xF2
#define portb     0x61
#define refresh_status 0x10

```

```

void fixdelay_15us ()
{
    // delay 15 us
    unsigned char char_ah,char_al ;
    char_ah = inportb ( portb ) & refresh_status ;
    fixdelay_loop :
    char_al = inportb ( portb ) & refresh_status ;
    if(char_ah == char_al ) goto fixdelay_loop ;
} // end of fixdelay_15us

```

```

main()
{
    unsigned char al_char ;

```

```

    printf("\n\n PORTWELL PPAP-3727 GPIO,3727GPIO.exe, V1.00 11-22-2006,All rights
reserved.\n\n");

```

```

printf("\n PPAP-3727   J30_Pin1=GPIO1:from SuperIO_GPIO17 ");
printf("\n          J30_Pin2=GPIO2:from SuperIO_GPIO16 ");
printf("\n          J30_Pin3=GPIO3:from SuperIO_GPIO15 ");
printf("\n          J30_Pin4=GPIO4:from SuperIO_GPIO14 ");
printf("\n          J30_Pin6=GPIO5:from SuperIO_GPIO10 ");
printf("\n          J30_Pin7=GPIO6:from SuperIO_GPIO11 ");
printf("\n          J30_Pin8=GPIO7:from SuperIO_GPIO12 ");
printf("\n          J30_Pin9=GPIO8:from SuperIO_GPIO13 ");
printf("\n\n   Put 4 jumper cap on J20 pin header pin1-6, 2-7, 3-8,4-9.\n");
printf("\n\n   Be careful !!! J30 pin5 (GND), and pin10 (Vcc) can not be shorted\n");

```

```

printf("\n\n Ready ? If yes , then Press any key to start test .....");
getche() ;

```

```

// W83627THG Super IO GP10~13 , 23~26 are used for PPAP-3719 GPIO
// GP10~13 located at LD7 , GP23~26 located at LD8
// ***** First : define the Multiplexed pins --- start
outportb ( config_W83627 , 0x87 ) ; // enter config mode
outportb ( config_W83627 , 0x87 ) ;
fixdelay_15us();

```

```

// CR29Bit[7,6]_P[01] , Define the multiplexed pin,GPIO10~17
outportb ( config_W83627 , CR29 ) ;
al_char = ( inportb ( config_W83627 +1 ) & 0x7D ) | 0x40 ;
outportb ( config_W83627 , CR29 ) ;
outportb ( config_W83627+1 , al_char ) ;

```

```

// Enable GPIO1 and GPIO2 and Set Non-inverse
outportb ( config_W83627 , CR07 );
outportb ( config_W83627+1 , GPIO1_LDN );

outportb ( config_W83627 , CR30 );
al_char = inportb ( config_W83627 +1 ) | 0x01 ;
outportb ( config_W83627 , CR30 );
outportb ( config_W83627+1 , al_char );

outportb ( config_W83627 , CRF2 );
outportb ( config_W83627+1 , 0x00 );
//LD8-----

```

/*

Testing way :

Use PPAP-3719 GPIO (8 bi-direction pins from Super IO W83627THG)
Initialization for W83627THG must be done first in Main().

--- t1

```

SGPO17 Write 0 to SGPI10 , SGPI10 = 0 ? ,if yes, pass ; if no, failed
SGPO16 Write 0 to SGPI11 , SGPI11 = 0 ? ,if yes, pass ; if no, failed
SGPO15 Write 0 to SGPI12 , SGPI12 = 0 ? ,if yes, pass ; if no, failed
SGPO14 Write 0 to SGPI13 , SGPI13 = 0 ? ,if yes, pass ; if no, failed

```

--- t2

```

SGPO17 Write 1 to SGPI10 , SGPI10 = 1 ? ,if yes, pass ; if no, failed
SGPO16 Write 1 to SGPI11 , SGPI10 = 1 ? ,if yes, pass ; if no, failed
SGPO15 Write 1 to SGPI12 , SGPI10 = 1 ? ,if yes, pass ; if no, failed
SGPO14 Write 1 to SGPI13 , SGPI10 = 1 ? ,if yes, pass ; if no, failed

```

--- t3

```

SGPO10 Write 0 to SGPI17 , SGPI26 = 0 ? ,if yes, pass ; if no, failed
SGPO11 Write 0 to SGPI16 , SGPI25 = 0 ? ,if yes, pass ; if no, failed
SGPO12 Write 0 to SGPI15 , SGPI24 = 0 ? ,if yes, pass ; if no, failed
SGPO13 Write 0 to SGPI14 , SGPI23 = 0 ? ,if yes, pass ; if no, failed

```

--- t4

```

SGPO10 Write 1 to SGPI17 , SGPI26 = 1 ? ,if yes, pass ; if no, failed
SGPO11 Write 1 to SGPI16 , SGPI25 = 1 ? ,if yes, pass ; if no, failed
SGPO12 Write 1 to SGPI15 , SGPI24 = 1 ? ,if yes, pass ; if no, failed
SGPO13 Write 1 to SGPI14 , SGPI23 = 1 ? ,if yes, pass ; if no, failed

```

*/

// GPIO Direction setting for t1 and t2 test items =====

```

outportb ( config_W83627 , CR07 );
outportb ( config_W83627+1 , GPIO1_LDN );

```

```

outportb ( config_W83627 , CRF0 );
outportb ( config_W83627+1 , 0x0F ); // Input direction ; SGPI10~13
                                   // Output direction ; SGPO17~14

```

// t1 -----

```

outportb ( config_W83627 , CR07 ) ;
outportb ( config_W83627+1 , GPIO1_LDN ) ;

outportb ( config_W83627 , CRF1 ) ;
outportb ( config_W83627+1 , 0x0F ) ;    // Write 0 ,

outportb ( config_W83627 , CRF1 ) ;
al_char = inportb ( config_W83627+1 ) & 0x0F ;

if( al_char == 0x00 ) goto next_t2 ;
outportb ( config_W83627 , 0xaa ) ; // exit config mode
printf("\n\n Error#01 : PPAP-3727 GPIO test failed(GPO17->GPI10_w0x0). But=0x%X\n\n",al_char);
exit(1);

// t2 -----
next_t2:
outportb ( config_W83627 , CRF1 ) ;
outportb ( config_W83627+1 , 0xFF ) ;    // Write 1 ,

outportb ( config_W83627 , CRF1 ) ;
al_char = inportb ( config_W83627+1 ) & 0x0F ;

if( al_char == 0x0F ) goto next_t3 ;
outportb ( config_W83627 , 0xaa ) ; // exit config mode
printf("\n\n Error#02 : PPAP-3727 GPIO test failed(GPO17->GPI10_w0xF). But=0x%X\n\n",al_char);
exit(1);

// GPIO Direction setting for t3 and t4 test items =====
next_t3:
outportb ( config_W83627 , CRF0 ) ;
outportb ( config_W83627+1 , 0xF0 ) ; // Output direction ; SGPO10~13
                                   // Input direction , SGPI17~14
// t3 -----
outportb ( config_W83627 , CRF1 ) ;
outportb ( config_W83627+1 , 0xF0 ) ;    // Write 0 ,

outportb ( config_W83627 , CRF1 ) ;
al_char = inportb ( config_W83627+1 ) & 0xF0 ;

if( al_char == 0x00 ) goto next_t4 ;
outportb ( config_W83627 , 0xaa ) ; // exit config mode
printf("\n\n Error#03 : PPAP-3727 GPIO test failed(GPO10->GPI17_w0x0). But=0x%X\n\n",al_char);
exit(1);

// t4 -----
next_t4:
outportb ( config_W83627 , CRF1 ) ;
outportb ( config_W83627+1 , 0xFF ) ;    // Write 1 ,

outportb ( config_W83627 , CRF1 ) ;

```

```

al_char = inportb ( config_W83627+1 ) & 0xF0 ;

if( al_char == 0xF0 ) goto gpio_done ;

outportb ( config_W83627 , 0xaa ) ; // exit config mode
printf("\n\n Error#04 : PPAP-3727 GPIO test failed(GPO10->GPI17_w0xF). But=0x%X
\n\n",al_char);
exit(1) ; //

gpio_done :

outportb ( config_W83627 , 0xaa ) ; // exit config mode
printf("\n\n PPAP-3727 GPIO test okay.  ^_^ \n\n");

exit(0);

} // end of main()

```

4.5 WDT Sample Code

DOSSEG

```
;-----  
; macro definition  
;  
;-----
```

;Purpose : To enter DOS service routines.

;Input : None.

;Output :

```
@DosCall    MACRO  FUNC  
    IFNB  <FUNC>  
    mov   ah, FUNC  
    ENDIF  
    int   21H  
    ENDM
```

;Purpose : To set address of interrupt handler.

;Input : InterruptNo - interrupt number

; EntryPoint - address of interrupt handler

;Output :

```
@SetIntvector    MACRO  InterruptNo,EntryPoint  
    IFNB  <InterruptNo>  
    mov   al,InterruptNo  
    ENDIF  
    IFNB  <EntryPoint>  
    lds   dx,EntryPoint  
    ENDIF  
    @DosCall 25H  
    ENDM
```

.MODEL small ; tiny

.STACK 200h

.DATA

```
;  
; ***** History *****  
;  
; V1.00 : 01042000 Release  
; V1.01 : 04212000 Release : Modify the display string " For ROBO-658  
;          SBC Card only " and the Jumper setting reminder .  
; V1.02 : 09052000 Release : Add ROBO-678 in support list  
; V1.03 : 08062001 Release : Add ROBO-616 , ROBO-679 , .....  
; V1.04 : 07032003 Release : Add ROBO-8820, PPAP-3723, .....  
;          Add WDT init, refresh 1000 time okay string  
;          and check W83627 onboard or not ?  
;          Strap 2E/2F, 4E/4F are tested.  
; V1.05 : 03282005 Release : Add checking Device ID = 82h in CR20 for 83627THF  
; V1.06 : 06144005 Release : HF , THF Multiplex pin ( Pin89 ) for WDTO are initied from  
;          different register bit.  
;          627HF --> CR2B_Bit4P0  
;          627THF --> CR2B_bit[3,2]P[0,1]
```

```
;PROMP1 DB'PORTWELL,INC. WDT Test ,W627V105, V1.05, 03-28-2005, All rights reserved.$'
PROMP1 DB'PORTWELL,INC. WDT Test, W627V106.EXE , V1.06, 06-14-2005, All rights
reserved.$'
```

```
PROMP1A DB ' !!!!! TEST PROGRAM FOR MS-DOS ENVIRONMENT !!!!! ',13,10,'$'
PROMP1B DB ' For the CPU Boards that have W83627HF/THF WDT design implemented on
board.',13,10,'$'
PROMP1C DB ' ',10,13,'$'
PROMP1D DB ' Please make sure WDT pin is output to a reset signal directly or
indirectly.',13,10,'$'
PROMP1E DB ' *****',13,10,'$'
PROMP1F DB ' * For examples : ',13,10,'$'
PROMP1G DB ' * For ROBO-658 : Set JP16 as "short" ',13,10,'$'
PROMP1H DB ' * For ROBO-678 : Set JP7 as "short" ',13,10,'$'
PROMP1I DB ' * For ACTI-777 : Set JP4 as "short" ',13,10,'$'
PROMP1J DB ' * For ROBO-616 : Set JP1 as "short" ',13,10,'$'
PROMP1K DB ' * RUBY-9712; PPAP-3711/3716/3719/3723... ',13,10,'$'
PROMP1L DB ' * ROBO-679/8710/8712/8713/8718/8820/8910/8911.....
*',13,10,'$'
PROMP1M DB ' *****',13,10,'$'
PROMP2 DB '< 1 > : ENABLE WDT, RREFRESH WDT 1000 TIMES,& DISABLE WDT !$'
PROMP3 DB '< 2 > : ENABLE WDT ONLY . ( This will ISSUE $'
PROMP3A DB 'system RESET when WDT expires ) $'
PROMP4 DB '< 3 > : EXIT.$'
PROMP5 DB ' Your choice : $'
PROMP6 DB ' Enable WDT at Te=$'
PROMP7 DB ' Refresh WDT : $'
PROMP8 DB ' Times at Ti=$'
PROMP9 DB ' Disable WDT !! $'
PROMPA DB ' THIS WILL RESET SYSTEM AFTER 1 second !!!$'
PROMPB DB ' EXIT WD TIMER TEST !$'
PROMPC DB 0DH,0AH,0DH,0AH,'$'
PROMPCC DB 0DH,0AH,'$'
PROMPDD DB ' second/minute : second ==> Twd : Retrigger interval ', '$'
PROMPD DB '< 1 > : 2 seconds: 0.3846 second .'
DB ' < A >: 40 seconds : 34.011 seconds . ',10,13,'$'
PROMPE DB '< 2 > : 3 seconds: 1.3736 seconds.'
DB ' < B >: 48 seconds : 40.824 seconds . ',10,13,'$'
PROMPF DB '< 3 > : 4 seconds: 2.363 seconds.'
DB ' < C >: 60 seconds : 51.044 seconds . ',10,13,'$'
PROMPG DB '< 4 > : 8 seconds: 5.604 seconds.'
DB ' < D >:120 seconds : 102.088 seconds . ',10,13,'$'
PROMPGA DB '< 5 >: 10 seconds: 7.033 seconds.'
DB ' < E >: 4 minutes : 204.176 seconds . ',10,13,'$'
PROMPGB DB '< 6 >: 16 seconds: 13.736 seconds.'
DB ' < F >: 8 minutes : 408.352 seconds . ',10,13,'$'
PROMPGC DB '< 7 >: 20 seconds: 17.033 seconds.'
DB ' < G >: 10 minutes : 510.439 seconds . ',10,13,'$'
PROMPGD DB '< 8 >: 24 seconds: 20.330 seconds.'
DB ' < H >: 30 minutes : 1531.318 seconds . ',10,13,'$'
PROMPGE DB '< 9 >: 32 seconds: 27.198 seconds.'
DB ' < I >: 60 minutes : 3060.000 seconds . ', '$'
PROMPG1 DB '< Q >: RETURN TO MAIN MENU. Ts=$'
PROMPH DB ' <<<<<<* OTHER CHOICE WILL NOT DELAY *>>>>>>.$'
```


PROMPI DB ' PRESS "Esc" KEY TO STOP REFRESHING WDT & DISABLE WDT.',10,13,'\$'

PROMPJ DB ' Program is running , please wait.....\$'

PROMP_NO627 DB 0dh,0ah,' ***** There is no W83627HF onboard *****',0dh,0ah,'\$'

PROMP_OK DB 'Refresh WDT 1000 times "PASS" (No reset),Hit any key for more test...','\$'
.CODE

old_int8 Dd ? ; old timer interrupt

count Dw 0

programstart:

mov ax,@data

mov ds,ax

; First initialization start -----

call WDT_enter ;

mov dx,2eh ; Read CR20

mov al,20h

out dx,al

mov dx,2fh

in al,dx

cmp al,52h ; W83627HF exist?

je W83627HF_EXIST

mov dx,4eh ; Read CR20

mov al,20h

out dx,al

mov dx,4fh

in al,dx

cmp al,52h ; W83627HF exist?

je W83627HF_EXIST

mov dx,2eh ; Read CR20

mov al,20h

out dx,al

mov dx,2fh

in al,dx

cmp al,82h ; W83627HF exist?

je W83627THF_EXIST

mov dx,4eh ; Read CR20

mov al,20h

out dx,al

mov dx,4fh

in al,dx

cmp al,82h ; W83627HF exist?

je W83627THF_EXIST

NO_W83627HF :

lea dx,PROMP_NO627

mov ah,9

int 21h

call WDT_exit

mov ah,4ch ; Return to DOS
int 21h

W83627HF_EXIST:

 ; 83627HF , CR2B_Bit4_P0
;----- 1-1 -----

mov dx,2eh ; Read CR2B
mov al,2bh
out dx,al
mov dx,2fh
in al,dx
and al,0efh ; CR2B_Bit4_P0
mov ah,al

mov dx,2eh
mov al,2bh
out dx,al
mov dx,2fh
mov al,ah
out dx,al

;----- 1-1 -----

;----- 1-2 -----

mov dx,4eh ; Read CR2B
mov al,2bh
out dx,al
mov dx,4fh
in al,dx
and al,0efh ; CR2B_Bit4_P0
mov ah,al

mov dx,4eh
mov al,2bh
out dx,al
mov dx,4fh
mov al,ah
out dx,al

;----- 1-2 -----

jmp W83627_EXIST

W83627THF_EXIST:

 ; 83627THF , CR2B_Bit[3,2]_P[0,1]
;----- 1-1 -----

mov dx,2eh ; Read CR2B
mov al,2bh
out dx,al
mov dx,2fh

```

in al,dx
and al,0F7h ; CR2B_Bit[3,2]_P[0,1]
or al,04h
mov ah,al

```

```

mov dx,2eh
mov al,2bh
out dx,al
mov dx,2fh
mov al,ah
out dx,al
;----- 1-1 -----

```

```

;----- 1-2 -----
mov dx,4eh ; Read CR2B
mov al,2bh
out dx,al
mov dx,4fh
in al,dx
and al,0F7h ; CR2B_Bit[3,2]_P[0,1]
or al,04h
mov ah,al

```

```

mov dx,4eh
mov al,2bh
out dx,al
mov dx,4fh
mov al,ah
out dx,al
;----- 1-2 -----

```

W83627_EXIST:

```

;----- 2-1 -----
mov dx,2eh ; Point to LDN8
mov al,07h
out dx,al
mov dx,2fh
mov al,08h
out dx,al

```

```

mov dx,2eh ; Read CR30
mov al,30h
out dx,al
mov dx,2fh
in al,dx
or al,01h
mov ah,al

```

```

mov dx,2eh ; Read CR30_Bit0_P1
mov al,30h

```

```
out dx,al
mov dx,2fh
mov al,ah
out dx,al
```

```
mov dx,2eh ; Read CRF5
mov al,0F5h
out dx,al
mov dx,2fh
in al,dx
and al,0FBh
mov ah,al
```

```
mov dx,2eh ; Read CRF5_Bit2_P0
mov al,0F5h
out dx,al
mov dx,2fh
mov al,ah
out dx,al
```

```
mov dx,2eh ; Read CRF7
mov al,0F7h
out dx,al
mov dx,2fh
in al,dx
and al,3Fh
mov ah,al
```

```
mov dx,2eh ; Read CRF7_Bit[7,6]_P[0,0]
mov al,0F7h
out dx,al
mov dx,2fh
mov al,ah
out dx,al
```

;----- 2-1 -----

;----- 2-2 -----

```
mov dx,4eh ; Point to LDN8
mov al,07h
out dx,al
mov dx,4fh
mov al,08h
out dx,al
```

```
mov dx,4eh ; Read CR30
mov al,30h
out dx,al
mov dx,4fh
in al,dx
or al,01h
mov ah,al
```

```
mov dx,4eh ; Read CR30_Bit0_P1
mov al,30h
out dx,al
```

```
mov dx,4fh
mov al,ah
out dx,al
```

```
mov dx,4eh ; Read CRF5
mov al,0F5h
out dx,al
mov dx,4fh
in al,dx
and al,0FBh
mov ah,al
```

```
mov dx,4eh ; Read CRF5_Bit2_P0
mov al,0F5h
out dx,al
mov dx,4fh
mov al,ah
out dx,al
```

```
mov dx,4eh ; Read CRF7
mov al,0F7h
out dx,al
mov dx,4fh
in al,dx
and al,3Fh
mov ah,al
```

```
mov dx,4eh ; Read CRF7_Bit[7,6]_P[0,0]
mov al,0F7h
out dx,al
mov dx,4fh
mov al,ah
out dx,al
```

```
;----- 2-2 -----
```

```
call WDT_exit
```

```
; First initialization end -----
```

```
init:
```

```
mov cx,25
mov ah,2
```

```
ccls: mov dl,0dh
```

```
int 21h
```

```
mov dl,0ah
```

```
int 21h
```

```
loop ccls
```

```
mov ah,2 ;set cursor position
```

```
mov bh,0
```

```
mov dh,0
```

```
mov dl,0
```

int 10h

```
lea dx,PROMP1
mov ah,9
int 21h
lea dx,PROMPC
int 21h
lea dx,PROMP1A
int 21h
lea dx,PROMP1B
int 21h
lea dx,PROMP1C
int 21h
lea dx,PROMP1D
int 21h
lea dx,PROMP1E
int 21h
lea dx,PROMP1F
int 21h
lea dx,PROMP1G
int 21h
lea dx,PROMP1H
int 21h
lea dx,PROMP1I
int 21h
lea dx,PROMP1J
int 21h
lea dx,PROMP1K
int 21h
lea dx,PROMP1L
int 21h
lea dx,PROMP1M
int 21h
lea dx,PROMPCC
int 21h
```

```
lea dx,PROMP2
int 21h
lea dx,PROMPC
int 21h
```

```
lea dx,PROMP3
int 21h
lea dx,PROMP3A
int 21h
lea dx,PROMPC
int 21h
```

```
lea dx,PROMP4
int 21h
lea dx,PROMPC
int 21h
lea dx,PROMPCc
int 21h
```

```

lea dx,PROMP5
int 21h

```

```

mov ah,1          ;read keyboard input
int 21h
cmp al,31h        ;choice 1
je choice1
cmp al,33h        ;choice 3
je choice3
cmp al,32h        ;choice 2
je choice2
jmp init

```

```

choice2:          ;RESET SYSTEM

```

```

mov ah,2
mov dl,7
int 21h
lea dx,PROMPC
mov ah,9
int 21h
lea dx,PROMPA
int 21h
lea dx,PROMPC
int 21h

```

```

; mov dx,Enable_port
; in al,dx
; jmp $+2

```

```

=====
=====

```

```

call WDT_enter
mov dx,0100h
call WDT_enable
call WDT_exit
mov ah,4ch
int 21h

```

```

choice3:          ;return to dos

```

```

mov ah,9
lea dx,PROMPC
int 21h
lea dx,PROMPB
int 21h
lea dx,PROMPC
int 21h

```

```

mov ah,4ch
int 21h

```

```

choice1:          ; enable wd , retrigger wd 1000 times & then disable wd

```

```

mov cx,26
mov ah,2
dcls: mov dl,0dh
int 21h
mov dl,0ah
int 21h

```



```

loop dcls
mov ah,2      ;set cursor position
mov bh,0
mov dh,0
mov dl,0
int 10h

```

```

mov ah,9
lea dx,PROMPC
int 21h
lea dx,PROMPDD
int 21h
lea dx,PROMPC
int 21h
lea dx,PROMPD
int 21h
lea dx,PROMPE
int 21h
lea dx,PROMPF
int 21h
lea dx,PROMPG
int 21h
lea dx,PROMPGA
int 21h
lea dx,PROMPGB
int 21h
lea dx,PROMPGC
int 21h
lea dx,PROMPGD
int 21h
lea dx,PROMPGE
int 21h
lea dx,PROMPC
int 21h
lea dx,PROMPG1
int 21h

```

```

;=====

```

```

;   mov ah,2
;   mov dl,0
;   int 21h      ; display space
;   int 21h

```

```

mov ah,2
int 1ah
mov al,ch      ; hours
call displayal
mov ah,2
mov dl,3ah     ; ':'
int 21h
mov al,cl      ; minutes
call displayal
int 21h       ; ':'
mov al,dh
call displayal ; seconds

```

```

;=====*****
;=====

    mov ah,2
    mov dl,0
    int 21h    ; display space
    int 21h
    int 21h

    mov ah,4
    int 1ah
    mov al,dh

    call displayal ; display month
    mov dh,dl
    mov ah,2
    mov dl,2fh    ; '/'
    int 21h
    mov al,dh

    call displayal ; day
    int 21h
    mov al,ch

    call displayal ; century
    mov al,cl

    call displayal ; year

;=====*****
;=====

    mov ah,9
    lea dx,PROMPC
    int 21h
    lea dx,PROMPH
    int 21h
    lea dx,PROMPC
    int 21h
    lea dx,PROMP5
    int 21h

    mov ah,1    ;read keyboard input
    int 21h    ; which choice
    cmp al,51h  ;return
    je going
    cmp al,71h
    je going
    cmp al,1bh
    jne ch124

```

```

;      jne chn2
going: jmp init
ch124: cmp al,31h      ;<1>  ;<1> --> BX=17   , <A> --> BX=680
      jne chn1          ;<2> --> BX=34   , <B> --> BX=816
      mov bx,7           ;<3> --> BX=68   , <C> --> BX=1050
      mov dx,0200h
      jmp chf            ;<4> --> BX=140   , <D> --> BX=2040
chn1:  cmp al,32h      ;<2>  ;<5> --> BX=170   , <E> --> BX=4080
      jne chn2          ;<6> --> BX=280   , <F> --> BX=8160
      mov bx,25         ;<7> --> BX=340   , <G> --> BX=10500
      mov dx,0300h
      jmp chf            ;<8> --> BX=408   , <H> --> BX=31500
chn2:  cmp al,33h      ;<3>  ;<9> --> BX=560   , <I> --> BX=63000
      jne chn3
      mov bx,43          ; others --> BX=31h
      mov dx,0400h
      jmp chf
chn3:  cmp al,34h      ;<4>
      jne chn4
      mov bx,102
      mov dx,0800h
      jmp chf
chn4:  cmp al,35h      ;<5>
      jne chn5
      mov bx,128
      mov dx,0a00h
      jmp chf
chn5:  cmp al,36h      ;<6>
      jne chn6
      mov bx,250
      mov dx,1000h
      jmp chf
chn6:  cmp al,37h      ;<7>
      jne chn7
      mov bx,310
      mov dx,1400h
      jmp chf
chn7:  cmp al,38h      ;<8>
      jne chn8
      mov bx,370
      mov dx,1800h
      jmp chf
chn8:  cmp al,39h      ;<9>
      jne chn9
      mov bx,495
      mov dx,2000h
      jmp chf
chn9:  cmp al,41h      ;<A>
      jne chna1
      mov bx,619
      mov dx,2800h
      jmp chf
chna1: cmp al,61h      ;
      jne chna2
      mov bx,619

```

```

    mov dx,2800h
    jmp chf
chna2: cmp al,42h    ;<B>
    jne chna3
    mov bx,743
    mov dx,3000h
    jmp chf
chna3: cmp al,62h    ;
    jne chna4
    mov bx,743
    mov dx,3000h
    jmp chf
chna4: cmp al,43h    ;<C>
    jne chna5
    mov bx,929
    mov dx,3c00h
    jmp chf
chna5: cmp al,63h    ;
    jne chna6
    mov bx,929
    mov dx,3c00h
    jmp chf
chna6: cmp al,44h    ;<D>
    jne chna7
    mov bx,1858
    mov dx,7800h
    jmp chf
chna7: cmp al,64h    ;
    jne chna8
    mov bx,1858
    mov dx,7800h
    jmp chf
chna8: cmp al,45h    ;<E>
    jne chna9
    mov bx,3716
    mov dx,0401h
    jmp chf
chna9: cmp al,65h    ;
    jne chnaa
    mov bx,3716
    mov dx,0401h
    jmp chf
chnaa: cmp al,46h    ;<F>
    jne chnab
    mov bx,7432
    mov dx,0801h
    jmp chf
chnab: cmp al,66h    ;
    jne chnac
    mov bx,7432
    mov dx,0801h
    jmp chf
chnac: cmp al,47h    ;<G>
    jne chnad
    mov bx,9290

```

```

        mov dx,0a01h
        jmp chf
chnad: cmp al,67h      ;
        jne chnae
        mov bx,9290
        mov dx,0a01h
        jmp chf
chnae: cmp al,48h      ;<H>
        jne chnaf
        mov bx,27870
        mov dx,1e01h
        jmp chf
chnaf: cmp al,68h      ;
        jne chnag
        mov bx,27870
        mov dx,1e01h
        jmp chf
chnag: cmp al,49h      ;<I>
        jne chnah
        mov bx,55692
        mov dx,3c01h
        jmp chf
chnah: cmp al,69h      ;
        jne chnai
        mov bx,55692
        mov dx,3c01h
        jmp chf
chnai: mov bx,31h
        mov dx,0200h
;        jmp chf
chf:

; ===== push WDT count 0 =====
        push dx      ;,,,,,, <----- Be careful

; ===== push WDT count 0 end=====

```

```

        mov ah,9
        lea dx,PROMPC
        int 21h
        lea dx,PROMP6      ; enable wd
        int 21h

```

```

;=====
;        mov ah,2
;        mov dl,0
;        int 21h      ; display space
;        int 21h

        mov ah,2
        int 1ah
        mov al,ch      ; hours
        call displayal

```

```

mov ah,2
mov dl,3ah      ; ':'
int 21h
mov al,cl       ; minutes
call displayal
int 21h         ; ':'
mov al,dh
call displayal  ; seconds

```

```

;=====*****

```

```

;=====

```

```

mov ah,2
mov dl,0
int 21h      ; display space
int 21h
int 21h

```

```

mov ah,4
int 1ah
mov al,dh

```

```

call displayal ; display month
mov dh,dl
mov ah,2
mov dl,2fh    ; '/'
int 21h
mov al,dh

```

```

call displayal ; day
int 21h
mov al,ch

```

```

call displayal ; century
mov al,cl

```

```

call displayal ; year

```

```

;===== POP WDT count 0=====
pop dx ;;;;;; <----- Be careful

```

```

;===== POP WDT count 0 end=====

```

```

call WDT_enter ; Enable WDT

```

```

call WDT_disable

```

```

call WDT_exit

```

```

call WDT_enter

```

```

call WDT_enable

```

```

call WDT_exit

```

```

;=====*****
;===== push WDT count1=====
    push dx ;;;;;; <----- Be careful

;===== push WDT count1 end===

    mov ah,9
    lea dx,PROMPC
    int 21h
    lea dx,PROMPI ; press "Esc" key to stop retriggering wd
    int 21h ; & disable wd
    lea dx,PROMPJ
    int 21h
    lea dx,PROMPC
    int 21h

    mov cx,1 ; count 1000 times for WDT refresh
loop1: ; refresh WDT

;===== pop WDT count1=====
    pop dx ;;;;;; <----- Be careful

;===== pop WDT count1 end===

    call WDT_enter ; Retrigger WDT

    call WDT_disable

    call WDT_exit

    call WDT_enter

    call WDT_enable

    call WDT_exit

    cmp bl,31h ; ----> nodelay
    je nodelay

.***** fix delay *****
,
    cmp bx,0h
    jne sfs1
    push cx
    mov cx,900
    call fixdelay
    pop cx
    jmp nodelay
.*****
,
    sfs1:

.***** fix delay *****
,
    cmp bx,0ffffh
    jne sfs2

```



```

    push cx
    mov cx,1800
    call fixdelay
    pop cx
    jmp nodelay
;*****
;
sfs2:
;-----
; delaytime:          Old way to delaytime
;
;   push cx
;   mov ax,1
;   ts: mov cx,1000
;   tt: out 0edh,al
;   loop tt
;   inc ax             ; BX=700 DELAY 1.0577 SEC.
;   cmp ax,bx         ; BX=600 DELAY .7355 SEC.
;   jl ts             ; BX=350 DELAY .529 SEC.
;   pop cx            ; BX= 60 DELAY .09068 SEC.
;-----

;-----
; delaytime:
; Main program entry pointer
;
; <1> BX=2 ---> DELAY 109.89 MS
; <2> BX=10 ---> DELAY 549.45 MS
; <3> BX=15 ---> DELAY 824.17 MS
; <4> BX=18 ---> DELAY 989.01 MS
; <5> BX=100---> DELAY 5.4945 SECONDS
; <6> BX=200---> DELAY 10.989 SECONDS
; <7> BX=250---> DELAY 13.736 SECONDS
; <8> BX=270---> DELAY 14.835 SECONDS
; <9> BX=280---> DELAY 15.384 SECONDS
; <A> BX=364---> DELAY 20.000 SECONDS
; <B> BX=546---> DELAY 30.000 SECONDS
; <C> BX=728---> DELAY 40.000 SECONDS
; <D> BX=910---> DELAY 50.000 SECONDS
; <E> BX=1092--> DELAY 60.000 SECONDS
; <F> BX=2093--> DELAY 115.000 SECONDS
; <G> BX=4368--> DELAY 240.000 SECONDS
; <H> BX=9100--> DELAY 500.000 SECONDS
; <I> BX=9373--> DELAY 515.000 SECONDS

;   BX=31h --> No delay.It takes about 3 seconds to retrigger WD 1000 times.
;-----

;
; hook int 08h
;
    mov     al,8
    mov     di,Offset old_int8
    mov     si,Offset int8
    call    chgint

```

```

;
;      mov    cs:count,bx
waiting:
      cmp    count, 0
      jnz    waiting
;
;
;
; unhook int 08h
;
;      push ax
      push dx
      push ds
      @SetIntvector 8,cs:old_int8
      pop ds
      pop dx
      push ax
      mov cs:count,0
;*****
;
nodelay:          ; NO DELAY TAKES ABOUT 3 SECONDS TO RETRIGGER
                  ; WD 1000 TIMES .

; ===== push WDT count2=====
      push dx      ; <----- Be careful

; ===== push WDT count2 end===

      mov ah,2
      mov dl,0dh
      int 21h
      lea dx,PROMP7
      mov ah,9      ; retrigger wd
      int 21h

;*****
;
      mov ax,cx      ; display trigger ??? times
      cmp ax,1000
      jne n1
      mov ah,2      ; display 1000 times
      mov dl,31h
      int 21h
      mov dl,30h
      int 21h
      int 21h
      int 21h
      jmp ok1

n1:          ; display < 1000 times
      mov ah,2      ; display space
      mov dl,20h
      int 21h

      mov ax,cx
      mov dl,100

```

```

div dl
mov dh,ah
cmp al,0
jne a1      ; al !=0 jmp display digit
            ; al =0  display space
mov ah,2
mov dl,20h
int 21h
jmp n2
a1:         ;?x?? x!=0
add al,30h
mov ah,2
mov dl,al
int 21h

n2:         ;??x? display
mov ax,cx
mov dl,100
div dl
mov al,ah
xor ah,ah
mov dl,10
div dl      ;
mov dh,ah   ; dh:???x
cmp al,0    ; al:??x?
jne n3

mov ah,2    ; ??0?
cmp cx,100
jge a2
mov dl,20h
jmp a3
a2: mov dl,30h
a3: int 21h
jmp n4
n3:         ; ??x? x!=0
add al,30h
mov ah,2
mov dl,al
int 21h
n4:         ; ???x display
add dh,30h ;
mov ah,2
mov dl,dh
int 21h
ok1:
lea dx,PROMP8 ; times
mov ah,9
int 21h
,*****
,
push cx

;=====
;
; mov ah,2
; mov dl,0

```

```

; int 21h ; display space
; int 21h

mov ah,2
int 1ah
mov al,ch ; hours
call displayal
mov ah,2
mov dl,3ah ; ':'
int 21h
mov al,cl ; minutes
call displayal
int 21h ; ':'
mov al,dh
call displayal ; seconds

;=====*****
;=====

mov ah,2
mov dl,0
int 21h ; display space
int 21h
int 21h

mov ah,4
int 1ah
mov al,dh

call displayal ; display month
mov dh,dl
mov ah,2
mov dl,2fh ; '/'
int 21h
mov al,dh

call displayal ; day
int 21h
mov al,ch

call displayal ; century
mov al,cl

call displayal ; year

pop cx
;=====*****

mov ah,6 ; check "Esc" key pressed ?
mov dl,0ffh
int 21h
cmp al,1bh
je done ; if "Esc" pressed , goto done

inc cx ; check 1000 times ?

```

```

        cmp cx,1001
        jge n6
        jmp loop1
n6:
        mov ah,2
        mov dl,0dh
        int 21h

; ===== pop WDT count2=====
        pop dx      ;;;;;;;;; <----- Be careful

; ===== pop WDT count2 end===

        lea dx,PROMP9
        mov ah,9      ; disable wd
        int 21h

        call WDT_enter
        call WDT_disable
        call WDT_exit

; ----- Display 1000 times test okay ----- Frank 07032003
        lea dx,PROMPCC
        mov ah,9      ; Line Feed and CR
        int 21h

        lea dx,PROMP_OK
        mov ah,9      ; Display "PASS" 1000 times test
        int 21h

        xor al,al
WAIT_KB:
        mov ah,1
        int 21h

        cmp al,0
        je WAIT_KB
; ----- Display 1000 times test okay ----- Frank 07032003

        jmp choice1

done:      ; "Esc" key has been pressed

; ===== pop WDT count2=====
        pop dx      ;;;;;;;;; <----- Be careful

; ===== pop WDT count2 end===

        call WDT_enter      ; disable WDT
        call WDT_disable
        call WDT_exit

        jmp init

```

```

;-----
; WDT_enter
; Input : None
; Return
; None
;-----
WDT_enter PROC    near
    push dx
    push ax
    pushf

    mov dx,2eh
    mov al,87h
    out dx,al
    jmp $+2
    out dx,al

    mov dx,4eh    ; Do another if 627 strapped at 4E/4F, 07032003
    mov al,87h
    out dx,al
    jmp $+2
    out dx,al

    popf
    pop ax
    pop dx

    ret
WDT_enter ENDP

```

```

;-----
; WDT_enable    ( Enable or Retrigger WDT )
; Input : DH ; the Time-out period 1 ---> FFh
;        DL ; Selection 0 ---> second, and 1 ---> Minute
; Return : None
;-----
WDT_enable PROC    near
    push dx
    push ax
    push bx
    pushf

    mov bx,dx    ; keep input value

    mov dx,2eh    ; Point to LDN 8
    mov al,07h
    out dx,al

    mov dx,2fh    ;
    mov al,08h
    out dx,al

```

```

mov dx,2eh ; read CRF5 first
mov al,0f5h
out dx,al

```

```

mov dx,2fh
in al,dx

```

```

cmp bl,1 ; Twd unit = minute ?
jne second
or al,08h ; Twd unit = minute
jmp nee

```

second:

```

and al,0f7h ; Twd unit = second

```

```

nee : mov ah,al
mov dx,2eh
mov al,0f5h
out dx,al

```

```

mov dx,2fh
mov al,ah
out dx,al

```

```

mov dx,2eh ; Twd count
mov al,0f6h
out dx,al

```

```

mov dx,2fh
mov al,bh
out dx,al

```

; For Strap 4E/4F

```

mov dx,4eh ; Point to LDN 8
mov al,07h
out dx,al

```

```

mov dx,4fh ;
mov al,08h
out dx,al

```

```

mov dx,4eh ; read CRF5 first
mov al,0f5h
out dx,al

```

```

mov dx,4fh
in al,dx

```

```

cmp bl,1 ; Twd unit = minute ?
jne second1
or al,08h ; Twd unit = minute
jmp nee1

```

second1:

```

and al,0f7h ; Twd unit = second

```

```

nee1 : mov ah,al

```



```
mov dx,4eh
mov al,0f5h
out dx,al
```

```
mov dx,4fh
mov al,ah
out dx,al
```

```
mov dx,4eh ; Twd count
mov al,0f6h
out dx,al
```

```
mov dx,4fh
mov al,bh
out dx,al
```

```
popf
pop bx
pop ax
pop dx
```

```
ret
```

```
WDT_enable ENDP
```

```
;-----
; WDT_disable
; Input : None
; Return : None
;-----
```

```
WDT_disable PROC near
```

```
push dx
push ax
pushf
```

```
mov dx,2eh ; Point to LDN 8
mov al,07h
out dx,al
```

```
mov dx,2fh ;
mov al,08h
out dx,al
```

```
mov dx,2eh
mov al,0f5h ; Reset WDTO mode to "Second"
out dx,al
```

```
mov dx,2fh
mov al,00
out dx,al
```

```
mov dx,2eh ; Disable WDT
mov al,0f6h
```

```
out dx,al
```

```
mov dx,2fh  
mov al,00h  
out dx,al
```

```
mov dx,4eh ; Point to LDN 8  
mov al,07h  
out dx,al
```

```
mov dx,4fh ;  
mov al,08h  
out dx,al
```

```
mov dx,4eh  
mov al,0f5h ; Reset WDTO mode to "Second"  
out dx,al
```

```
mov dx,4fh  
mov al,00  
out dx,al
```

```
mov dx,4eh ; Disable WDT  
mov al,0f6h  
out dx,al
```

```
mov dx,4fh  
mov al,00h  
out dx,al
```

```
popf  
pop ax  
pop dx
```

```
ret
```

```
WDT_disable ENDP
```

```
-----  
; WDT_exit  
; Input : None  
; Return  
; None  
-----
```

```
WDT_exit PROC near
```

```
push dx  
push ax  
pushf
```

```
mov dx,2eh  
mov al,0aah  
out dx,al
```

```
mov dx,4eh  
mov al,0aah
```

```

    out dx,al

    popf
    pop ax
    pop dx

    ret
WDT_exit ENDP
;-----;
;          FIXED_DELAY          ;
;-----;
;   Input : (CX) count of 15 microseconds to wait      ;
;   STACK PRESENT                                         ;
;   Output: NONE                                           ;
;   Register destroyed : (CX)                             ;
;   ;
; This routine is called to wait for 15 microseconds * count in ;
; (CX), then return. Gives a programmed software delay.    ;
;-----;
FIXDELAY PROC    near
    push dx
    push ax
    pushf

    mov     dx,61h
    in      al,dx      ;
    jmp     $+2
    jmp     $+2
    and     al,00010000b ;
    mov     ah,al      ;
fixed_delay_1:
    in      al,dx      ;
    jmp     $+2
    jmp     $+2
    and     al,00010000b ;
    cmp     al,ah      ;
    jz      short fixed_delay_1 ;
    mov     ah,al      ;
    loop    short fixed_delay_1 ;

    popf
    pop ax      ;
    pop dx
    ret
FIXDELAY ENDP
;-----;
; Initialize and change the interrupt vector of INT
; Call
;   AL = INT number
;   DI = the offset to save old int vector
;   SI = the offset of new int
; Return
;   None
;-----;
CHGINT PROC    near

```

```

PUSH  AX
PUSH  BX
PUSH  ES

xor    bx,bx
mov    es,bx
xor    ah,ah
shl    ax,1
shl    ax,1
mov    bx,ax
mov    ax,es:[bx]
mov    cs:[di],ax
mov    ax,es:[bx+2]
mov    cs:[di+2],ax
cli
mov    ax,cs
mov    es:[bx],si
mov    es:[bx+2],ax
sti

POP ES
POP BX
POP AX
ret
CHGINT ENDP

```

```

;- INT8 -----
; Timer interrupt service routine
;
; Input   : None.
; Output  :
; Modified :
;-----

```

```

int8 Proc far

    cmp    cs:count,0      ;
    je     @F
    dec    cs:count
@F:
    jmp     Dword Ptr cs:old_int8

```

```

int8 Endp

```

```

displayal proc near ; display content in AL
    push dx
    push ax

    and al,0f0h      ; display high nibble
    shr al,1
    shr al,1
    shr al,1
    shr al,1

```

```

    cmp al,0ah
    jae aa4
    add al,30h
    jmp aa5
aa4:
    add al,37h
aa5:
    mov dl,al
    mov ah,2
    int 21h

    pop ax          ; display low nibble
    push ax
    and al,0fh


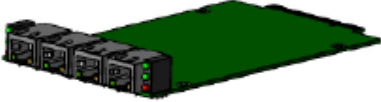

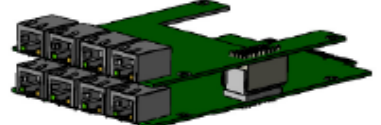


    cmp al,0ah
    jae aa6
    add al,30h
    jmp aa7
aa6:
    add al,37h
aa7:
    mov dl,al
    mov ah,2
    int 21h        ;
    pop ax
    pop dx
    ret
displayal endp


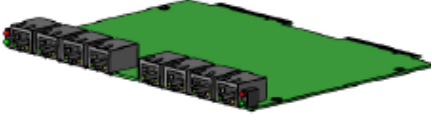


END programstart

```

5.1 NAR-7090 Ethernet modules configuration

This chapter will show what Ethernet modules that NAR-7090 supports. NAR-7090 have three slots to support PCI-E x8 module. The front panel may different when use these modules. Below is the list for all Ethernet modules.

				
Modules Name	Support Slots	Bypass Gen2 support	RJ-45(C) Fiber (F) Support	Ethernet Chip
ABN-454 	A, B, C	X	RJ-45 x4	Intel 82571 x2
ABN-464 	A, B, C	X	Fiber x4	Intel 82571 x2
ABN-458 	A, B, C	X	RJ-45 x8	Intel 82571 x4
ABN-482 	A, B, C		Fiber x2	Intel 82571 x1
ABN-484 	A, B, C		RJ-45 x4	Intel 82571 x2

ABN-476 	A+B		RJ-45 x4 Fiber x2	Intel 82571 x3
ABN-478 	A+B		RJ-45 x8	Intel 82571 x4
ABN-486 	A+B		Fiber x6	Intel 82571 x3
ABN-522 	A, B	X	10G Fiber x2	Intel 82598 x1